BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

FAULT-TOLERANT TIME-OUT COMMUNICATION PROTOCOL AND SENSOR APPARATUS FOR USING SAME

CROSS-REFERENCE TO RELATED APPLICATION

The present application claims the benefit of provisional patent application No. 60/450,000, filed February 25, 2003, which application is hereby incorporated by reference.

TECHNICAL FIELD

The present invention relates to sensors, e.g., sensor arrays and networks. More particularly, the present invention pertains to communication methods for use with such sensors.

SUMMARY OF THE INVENTION

The present invention provides various embodiments of communication protocols for use, for example, in communication of sensor information in system, sensor network, sensor array, etc.

In one embodiment of a method of communicating sensor information, the method includes providing a sensor architecture comprising a base station and a plurality of sensors associated with the base station. Each sensor is operable to communicate sensor information to the base station and backup information to one or more other sensors of the plurality of sensors. Failed communication of sensor information from at least one sensor of the plurality of sensors to the base station is detected, e.g., a time-out occurs. Task information representative of the failed communication is broadcast to the one or more other sensors receiving backup sensor information from the at least one sensor to request priority information therefrom. Backup sensor information is requested to be sent to the base station from one of the one or more other sensors receiving such backup sensor information based on the priority information received from each of the one or more other sensors.

In another embodiment of a method of communicating sensor information, the method includes providing a sensor architecture comprising a base station and a plurality of sensors associated with the base station. Each sensor is operable to communicate sensor information to the base station and backup information to one or more other sensors of the

plurality of sensors. Failed communication of sensor information from at least one sensor of the plurality of sensors to the base station is detected. Upon detection, an evaluation is performed of the potential ability of each of the one or more other sensors in providing backup sensor information to the base station. The backup sensor information is requested to be sent to the base station from one of the one or more other sensors based on the evaluation. For example, the evaluation maybe performed by considering the time needed for each sensor to transmit the backup sensor information to the base station and/or the number of tasks offered to each sensor.

The present invention contemplates the use of the sensor communication described herein in various sensor networks, apparatus, or systems. Hardware, programs and/or algorithms for use in providing the described above features according to the present invention are also contemplated.

The above summary of the present invention is not intended to describe each embodiment or every implementation of the present invention. Advantages, together with a more complete understanding of the invention, will become apparent and appreciated by referring to the following detailed description and claims. Various other embodiments of the present invention are provided by the below Description.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In the following detailed description of the embodiments, reference is made to specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and processing step/structural changes may be made without departing from the scope of the present invention. Further, the word "backup," when used, for example, in association with a sensor or node, may be interchangeable with the term "sibling."

The following "IV. Detailed Description of the Invention" provides various embodiments of the present invention. It will be recognized that various components or elements of the present invention and/or method steps carried out as described therein may be included and/or may be optional according to the present invention.

IV. Detailed Description of the Invention

Abstract:

Flow measurement is an important technology used in nearly every industry. Distributed micro flow-sensor arrays and networks (DMFSA/N), built from collections of spatially scattered, cooperating intelligent micro flow-sensor nodes, can improve the accuracy and reliability of the system. However, it is unrealistic to expect all the sensor nodes and communication links in the system to be fault-free all the time. A fault-tolerant time-out Protocol (FTTP) with two alternatives - one with blind station, and the other with smart base station was proposed in our research. Experiment results show that compared with FTTP with blind base station, although FTTP with smart base station requires more complex structures, it involves less communication.

I. Distributed Micro Flow-Sensor Arrays and Networks (DMFSA/N) Structure

Before FTTP is presented in detail, it is necessary to briefly describe the DMFSA/N structure that supports the communication among the nodes in the system. Figure 1 shows the clustered DMFSA/N structure. In this structure, DMFSA/N are divided into *sensor cluster units* (SCUs), each of which consists of a set of intelligent micro flow-sensor nodes and a base station. The intelligent sensor nodes within the same SCU are also called *sibling nodes*. An intelligent micro flow-sensor node consists of a dedicated processing unit and an associated micro flow-sensor that measures the variable of interest. A more powerful processing unit is used as the base station, which acts as the control node of the respective SCU, controlling the signal transmission of the sensor node, managing information rerouting in case of link failures and integrating the information from all the sensor nodes in the SCU. In order to make the system tolerant to link failures, the sensor node sends information not only to the base station, but also to one or more other sibling nodes that are also called its *backup nodes*. Two communication protocols can be used among the sibling nodes: (1) Broadcasting Protocol, in which each sensor node broadcasts its information to all its sibling nodes; (2) *Evenly wide gossiping protocol* (EWGP), a revised Gossiping Protocol (GP), in which instead of sending information to only one other sibling node by random selection according to GP, the backup nodes are chosen in the way that the roles of backup nodes are evenly distributed among the sibling nodes, so it will not happen that some nodes get too many 'duties', while others are 'starved'. These communication protocols can also be applied in the communication among the base station. The external processor is the commander of the entire network, in which information from all the SCUs is integrated for decision making as required. The clusters can be deployed along the flow path, depending on the flow situation and applications (Figure 1).

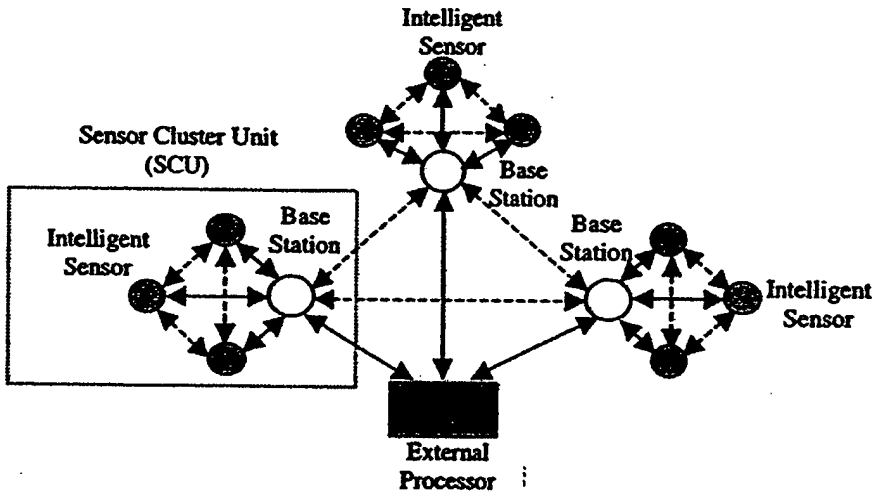


Figure 1. Clustered DMFSA/N Structure (dash line means the link may or may not exist, depending on the communication protocol used)

II. Fault-tolerant Time-out Communication Protocol

The design of fault-tolerant time-out protocol (FTTP) is motivated by the combination of time-out and task coordination protocols and the need for fault-tolerant integration. At the beginning of each data transmission cycle, the base station broadcasts the data transmission request to all the sensor nodes in the SCU. Receiving the request, each sensor node sends its data to the buffer inside the base station. Each sensor node has a corresponding buffer in the base station to store its data so that the base station can trace the source of the data. Then the data will be retrieved into the processing unit in the base station. According to FTTP, the base station will stop waiting for the information from the sensor node if a certain amount of time, T, has passed, where T is the pre-determined time-out period and its value depends on the application. Base station will then announce the rerouting task message to its sibling nodes. Two alternatives exist for where the base station should send the task announcement: (1) with a Blind Base Station, the message is sent to all the sibling nodes; (2) with a Smart Base Station, the message is only sent to the sibling node that can finish the task earliest, based on the current status of the nodes in the system.

FTTP with Blind Base Station

In this protocol, when the link of a sensor node, say node v_k, fails, the base station will broadcast the rerouting task message (RTM) for v_k to all its sibling nodes. Receiving RTM, its backup node b_{ki} (the ith backup node of v_k) generates a bid value bv_{ki} and sends it to the base station. $b_{ki} \in BK_k$, where BK_k is the set of backup nodes of v_k. bv_{ki} can be calculated as follows:

$$bv_{ki} = (m_{ki} + 1)\gamma_{ki} \qquad (1)$$

where γ_H is the transmission rate of node b_H (time spent in transmitting one message) and m_H is the number of tasks that have been offered to node b_H. Because it is unrealistic for the intelligent sensor node to keep track of its transmission rate over time due to its computation limit, γ_H is assumed to be constant for node b_H. The base station then evaluates the priorities of the backup nodes based on their bid values (the lower the bid value, the higher the priority), and sends the task offer message (TOM) to the backup node with the highest priority, $b_{H(1)}$. Receiving the task offer message, $b_{H(1)}$ sends the required backup data message (BDM) to the base station. The structures of the base station and intelligent nodes, and the logic chart of the FTTP with blind base station are shown in Figure 2 and 3 respectively.

FTTP with Smart Base Station

In this protocol, in order to limit the unnecessary communication and reduce the communication traffic, instead of broadcasting RTM for v_k, whose link fails, to all its sibling nodes, the base station first evaluates the priorities of the its backup nodes and then sends TOM only to the node that can finish the task the earliest. The time when the backup node b_H finishes the task, t_H, can be calculated as (assuming the transmission starts at time 0):

$$t_H = (m_H + 1)\gamma_H \qquad (2)$$

where m_H is the number of tasks that have been offered to the backup node b_H, and γ_H is the transmission rate of b_H (time spent in transmitting one message). The base station then sends TOM to the node $b_{H(1)}$ whose $t_{k(1)}$ is the smallest. In order to find out t_H, the base station needs to have a knowledge base which keeps track of the information for each node – its backup nodes, transmission rate, and the number of tasks that have been offered. Figure 4 shows the structures of base station and intelligent sensor nodes for the FTTP with smart base station. The model of the knowledge base is shown in Figure 5, in which n is the number of sensor nodes in the cluster and the backup nodes are hypothetical for illustration only. Figure 6 shows the logic chart of the FTTP with smart base station.

Compared with the FTTP with blind base station, the FTTP with smart base station requires less communication, and has the advantage of being able to keep track of the updated transmission rate of each sensor node, but it requires a base station with more complex structure (with an additional knowledge base). Therefore, a tradeoff has to be considered in order to determine when to apply each protocol.

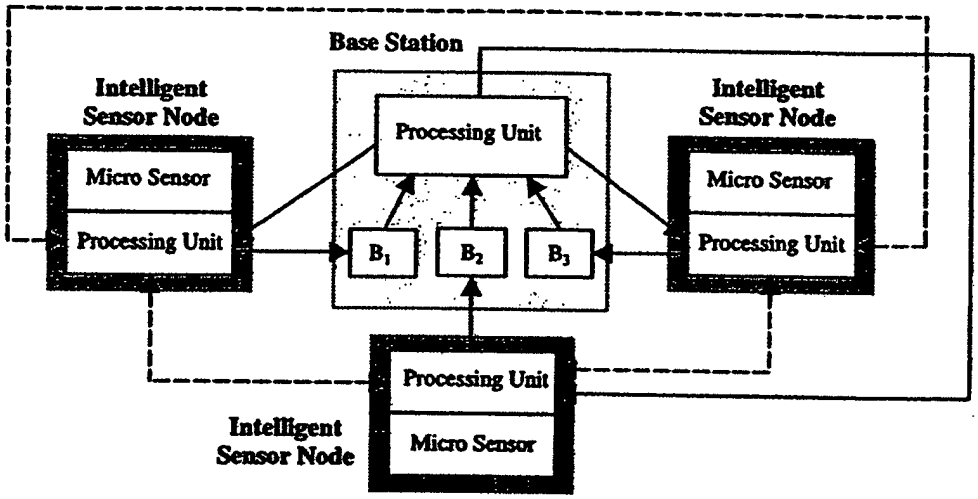


Figure 2. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Blind Base Station (Dash line means the link may or may not exist depending on the protocol used.)

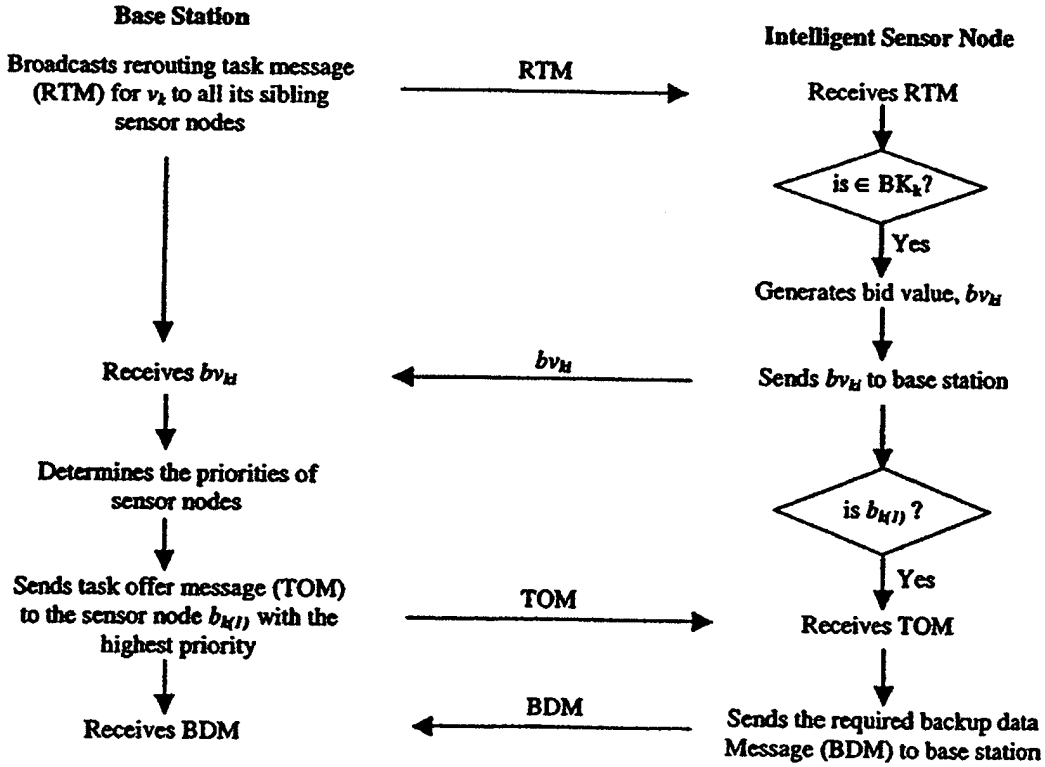


Figure 3. Logic Chart of the FTTP with Blind Base Station

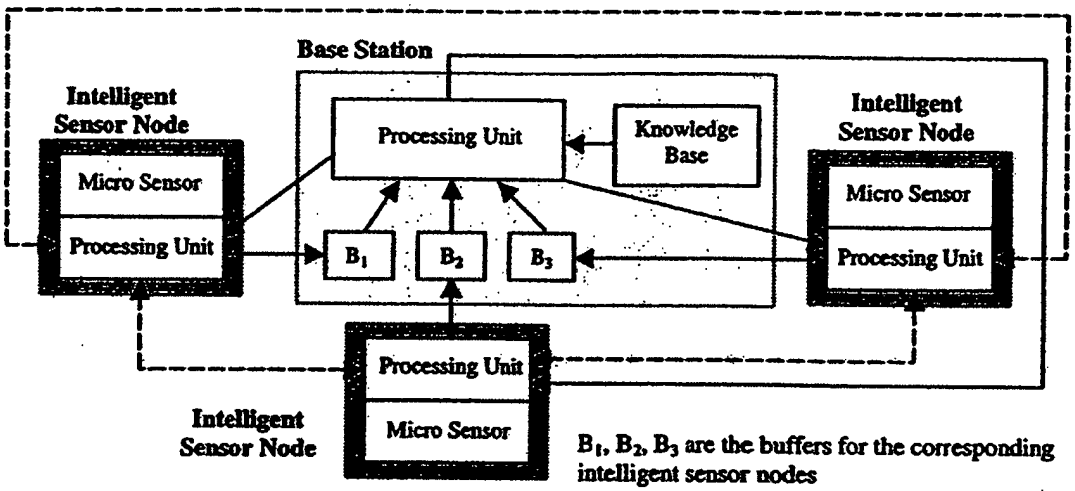
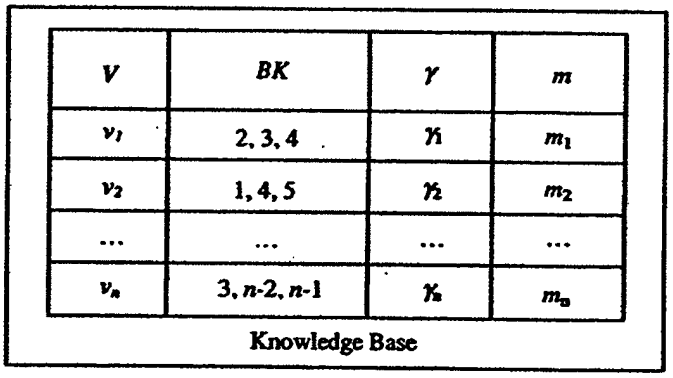


Figure 4. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Smart Base Station (Dash line means the link may or may not exist depending on the protocol used.)



V	BK	γ	m
v_1	2, 3, 4	γ_1	m_1
v_2	1, 4, 5	γ_2	m_2
...	...	...	...
v_n	3, n-2, n-1	γ_n	m_n

Knowledge Base

Figure 5. Logic Chart of the FTTP with Smart Base Station

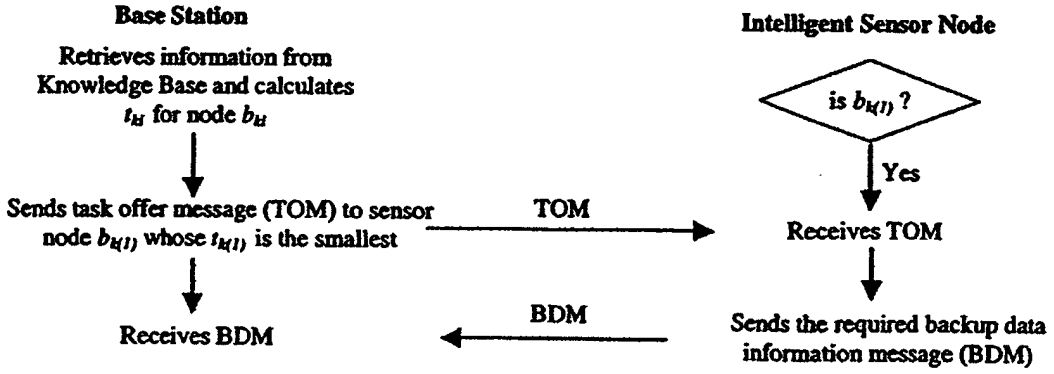


Figure 6. Logic Chart of the FTTP with Smart Base Station

(Fig. 3) Demo

fttpb.c

1 mpirun -np [# of processors] a.out

```
#include<stdio.h>
#include<stdlib.h>
#include"mpi.h"

#include "ev_bid.c" /* File used to evaluate the bid value */

#define DATA 1
#define BV 2
#define INFO 3
#define OFFER 4
#define SYNCHRON 5

void base_station();
void sensor_node();

int processors,myrank;

main(int argc, char *argv[])
{
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&processors);
  MPI_Comm_rank(MPI_COMM_WORLD,&myrank);

  if(myrank==0) {

        base_station();
  }

  else {

        sensor_node();
  }

   MPI_Finalize();
}

void base_station()
{
   double bid,bv[50],data,info,interval,start_e,interval_e,interval_t,interval_r,start
_s,interval_s,interval2,interval_b,start_b,temp;
   int ts,i,j,fail,winner,source,offer;
   MPI_Status status;
   FILE *f1;
   f1=fopen("fttpb.dat","a");

   /* Input the number of failed links */
   printf("How many link failures?\n");
   scanf("%d",&fail);
   MPI_Bcast(&fail,1,MPI_INT,0,MPI_COMM_WORLD);

   /* Initialization */
   interval_t=0.0;
   interval_b=0.0;
   interval_r=0.0;
   interval_e=0.0;
   interval_s=0.0;

   for (i=1;i<processors;i++) {
      MPI_Recv(&info,1,MPI_DOUBLE,MPI_ANY_SOURCE,INFO,MPI_COMM_WORLD,&status);
   }

   for (j=1;j<=fail;j++) {
      start_b=MPI_Wtime();
      MPI_Bcast(&ts,1,MPI_INT,0,MPI_COMM_WORLD);
      temp=MPI_Wtime()-start_b;/* Time for broadcasting the rerouting task message */
      interval_b=interval_b+temp;
      for (i=1;i<processors;i++) {
        MPI_Recv(&bid,1,MPI_DOUBLE,MPI_ANY_SOURCE,BV,MPI_COMM_WORLD,&status);/* Recei
ve the bid values */
         source=status.MPI_SOURCE;
         MPI_Recv(&interval,1,MPI_DOUBLE,source,SYNCHRON,MPI_COMM_WORLD,&status);
         bv[source]=bid;
         interval_r=interval_r+interval;/* Time for receiving the bid values from all
the sensor nodes */
      }
      start_e=MPI_Wtime();
      winner=evaluateBid(bv,processors);
      interval_e=interval_e+MPI_Wtime()-start_e;/* Time for evaluating the bid values
*/
      MPI_Bcast(&winner,1,MPI_INT,0,MPI_COMM_WORLD);
      start_s=MPI_Wtime();
      MPI_Send(&offer,1,MPI_INT,winner,OFFER,MPI_COMM_WORLD);
      temp=MPI_Wtime()-start_s;
      interval_s=interval_s+temp;
      MPI_Recv(&data,1,MPI_DOUBLE,winner,DATA,MPI_COMM_WORLD,&status);
      MPI_Recv(&interval2,1,MPI_DOUBLE,winner,SYNCHRON,MPI_COMM_WORLD,&status);
      interval_s=interval_s+interval2;/* Time for sending task offer and receiving
 required data */
   }
   interval_t=interval_b+interval_r+interval_e+interval_s; /* Total time */

   /* Print out the results */
   printf("evaluating time:%.6f, sending/receiving time:%.6f, total time:%.6f\n",inter
val_e, interval_b+interval_r+interval_s,interval_t);
   fprintf(f1,"# of sensor nodes:%d, # of failed links:%d, evaluating time:%.6f, sendi
ng/receiving time:%.6f, total time:%.6f\n",processors-1,fail,interval_e,interval_b+int
erval_r+interval_s,interval_t);
   fclose(f1);
}

void sensor_node()
{
   int ts,i,fail,jobNo,winner,offer;
   double data,bid,info,tranTime,start_t,interval,start_s,interval2;
   MPI_Status status;
   MPI_Bcast(&fail,1,MPI_INT,0,MPI_COMM_WORLD);

   start_t=MPI_Wtime();
   MPI_Send(&info,1,MPI_DOUBLE,0,INFO,MPI_COMM_WORLD);
   tranTime=MPI_Wtime()-start_t; /* Transmission time for each sensor node */
   jobNo=1;
   for (i=1;i<=fail;i++) {
      MPI_Bcast(&ts,1,MPI_INT,0,MPI_COMM_WORLD); /* Receive the broadcasted rerouting
task message */
      start_t=MPI_Wtime();
      bid=jobNo*tranTime; /* Compute bid value */
      MPI_Send(&bid,1,MPI_DOUBLE,0,BV,MPI_COMM_WORLD);/* Send bid value */
      interval=MPI_Wtime()-start_t;
      MPI_Send(&interval,1,MPI_DOUBLE,0,SYNCHRON,MPI_COMM_WORLD);/* Synchronize the se
nder and receiver */
      MPI_Bcast(&winner,1,MPI_INT,0,MPI_COMM_WORLD);
      if (myrank==winner) {
        MPI_Recv(&offer,1,MPI_INT,0,OFFER,MPI_COMM_WORLD,&status);/* Receive task off
er */
         start_s=MPI_Wtime();
         jobNo++;
         MPI_Send(&data,1,MPI_DOUBLE,0,DATA,MPI_COMM_WORLD);/* Send required data */
         interval2=MPI_Wtime()-start_s;
         MPI_Send(&interval2,1,MPI_DOUBLE,0,SYNCHRON,MPI_COMM_WORLD);
      }
   }
}
```

cc fttps.c -lmpi

1 mpirun -np [# of processors] a.out

fttps.c

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include"mpi.h"

#include "ev_bid.c" /* File used to evaluate the priorities of backup sensor nodes */

#define DATA 1
#define BV 2
#define OFFER 3
#define SYNCHRON 4
#define INFO 5
#define TRANSRATE 6

void base_station();
void sensor_node();

int processors,myrank;

main(int argc, char *argv[])
{
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&processors);
  MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
  if(myrank==0) {
       base_station();
  }
  else {
       sensor_node();
  }
   MPI_Finalize();
}

void base_station()
{
   double start_t,start_b,finish_b,finish_o,interval_e,interval_s,interval_r,interval,
interval_t,temp,tranTime[50], bv[50], jobNo[50],info,data,trans_t;
   int i,j,fail, winner,source,offer;
   MPI_Status status;
   FILE *f1;
   f1=fopen("fttps.dat","a");

   /* Input the number of failed links */
   printf("How many link failures?\n");
   scanf("%d",&fail);
   MPI_Bcast(&fail,1,MPI_INT,0,MPI_COMM_WORLD);

  /* Initialization */
   interval_t=0.0;
   interval_e=0.0;
   interval_s=0.0;
   interval_r=0.0;

   /* Get information(transmission rate and number of task offers) of each backup sens
or node */
   for (i=1;i<processors;i++) {
      MPI_Recv(&info,1,MPI_DOUBLE,MPI_ANY_SOURCE,INFO,MPI_COMM_WORLD,&status);
      source=status.MPI_SOURCE;
      MPI_Recv(&trans_t,1,MPI_DOUBLE,source,TRANSRATE,MPI_COMM_WORLD,&status);
      tranTime[source]=trans_t;
      jobNo[source]=1;
   }

   for (j=1;j<=fail;j++) {
      start_t=MPI_Wtime();
      for (i=1;i<processors;i++) {
         bv[i]=jobNo[i]*tranTime[i]; /* Compute time when each backup sensor node can
finish the rerouting task */
      }
      winner=evaluate(bv,processors);/* Evaluate the priorities of the sensor nodes
 */
      jobNo[winner]++; /* Update information of the selected sensor node */
      start_b=MPI_Wtime();
      interval_e=start_b-start_t+interval_e;/* Time for evaluating the priorites and u
pdating information of the sensor node */
      MPI_Bcast(&winner,1,MPI_INT,0,MPI_COMM_WORLD);
      finish_b=MPI_Wtime();
      MPI_Send(&offer,1,MPI_INT,winner,OFFER,MPI_COMM_WORLD);
      finish_o=MPI_Wtime();
      temp=finish_o-finish_b; /* Time for sending task offer */
      interval_s=interval_s+temp;
      MPI_Recv(&data,1,MPI_DOUBLE,winner,DATA,MPI_COMM_WORLD,&status);
      MPI_Recv(&interval,1,MPI_DOUBLE,winner,SYNCHRON,MPI_COMM_WORLD,&status);
      tranTime[winner]=interval;
      interval_r=interval+interval_r;/* Time for receiving the required data */

   }
      interval_t=interval_e+interval_s+interval_r;
      printf("evaluating time:%.6f,sending/receiving time:%.6f,total time:%.6f\n",inte
rval_e,interval_s+interval_r,interval_t);
      fprintf(f1,"# of sensor nodes:%d, # of failed links:%d, interval_t:%.6f,evaluati
ng:%.6f,sendingOffer:%.6f,recevingData:%.6f\n",processors-1,fail,interval_t,interval_e
,interval_s,interval_r);
      fclose(f1);
}

void sensor_node()
{
   int i,fail,winner;
   double start_t,start_s,interval,interval_s,data,info;

   MPI_Bcast(&fail,1,MPI_INT,0,MPI_COMM_WORLD);

   start_s=MPI_Wtime();
   MPI_Send(&info,1,MPI_DOUBLE,0,INFO,MPI_COMM_WORLD);
   interval_s=MPI_Wtime()-start_s;
   MPI_Send(&interval_s,1,MPI_DOUBLE,0,TRANSRATE,MPI_COMM_WORLD);
   for (i=1;i<=fail;i++) {
      MPI_Bcast(&winner,1,MPI_INT,0,MPI_COMM_WORLD);/* Receive task offer */
      if(myrank==winner) {
         start_t=MPI_Wtime();
         MPI_Send(&data,1,MPI_DOUBLE,0,DATA,MPI_COMM_WORLD); /* Send required data */
         interval=MPI_Wtime()-start_t;
         MPI_Send(&interval,1,MPI_DOUBLE,0,SYNCHRON,MPI_COMM_WORLD);/* Synchronize the
 sender and the receiver */
      }
   }
}
```

ev_bid.c

```
int evaluateBid(double *bv, int processors) {
   int i,j,winner;
   double temp;
   winner=1;

   for (i=2;i<processors;i++) {
         if (bv[1]>bv[i]) {
            temp=bv[1];
            bv[1]=bv[i];
            bv[i]=temp;
            winner=i;
         }
   }

   return winner;
}
```

The following "VII. Information" provides information regarding one or more applications for the present invention and problems that the present invention may be used to resolve.

VII. Information

Distributed micro flow-sensor arrays and networks (DMFSA/N) will open an original direction in the applications of flow measurement and leak detection. The ability to place the micro flow-sensor nodes when and where they are needed provides better control of process. However, It is unrealistic to expect all the micro sensor nodes and communication links along the path to be fault-free all the time because of the constraints of micro sensors and complex flow environment. In order to make the system link failure tolerant and satisfy the real-time control requirement, fault-tolerant time-out communication protocol (FTTP), which has two alternatives, is proposed. FTTP integrates the conventional time-out protocol with fault-tolerant information integration characteristics.

1. It addresses the fault-tolerance issue in the design of DMFSA/DMFSN
2. This protocol is easy to understand and implement.
3. It can easily be revised to cater to the applications similar to the DMFSA/DMFSN.

It addresses the issues in the design of distributed micro flow-sensor arrays and network (DMFSA/DMFSN) or other similar applications. Due to the physical constraints of the micro sensors and effects of the environment they are measuring, it is unrealistic to expect all the micro sensor nodes and communication links along the path to be fault-free all the time. In order to make the system link failure tolerant and satisfy the real-time control requirement, fault-tolerant time-out protocol, which has two alternatives, is proposed in our research.

In the past, similar problems were raised in the design of sensor networks. The most common approach was to increase the number of the connections among the sensor nodes or the number of sensor nodes.

The following materials entitled "Distributed Micro Flow-Sensor Network (DMFSN) Design and Modeling" provide generalized information about a potential communication protocol.

Distributed Micro Flow-Sensor Network (DMFSN) Design and Modeling

Topics

- MEMS, Micro Flow-Sensor, and DMFSN
- Design challenges of Micro Flow-Sensor and DMFSN
- Design of DMFSN Architecture
- Design of Information Communication Protocols for DMFSN
- Teamwork Integration Evaluator(TIE)/MEMS
- Conclusions

MEMS and Micro Flow-Sensor

- Micro Electro Mechanical System (MEMS)
  - deals with designing and fabricating entire electrical and mechanical systems, usually on a single silicon chip
- Micro Flow-Sensors
  - Interfere less with the environment
  - Lower manufacturing cost
  - Can be applied in narrow/tiny spaces
  - Can be used with redundancy to improve information accuracy and fault tolerance of the system

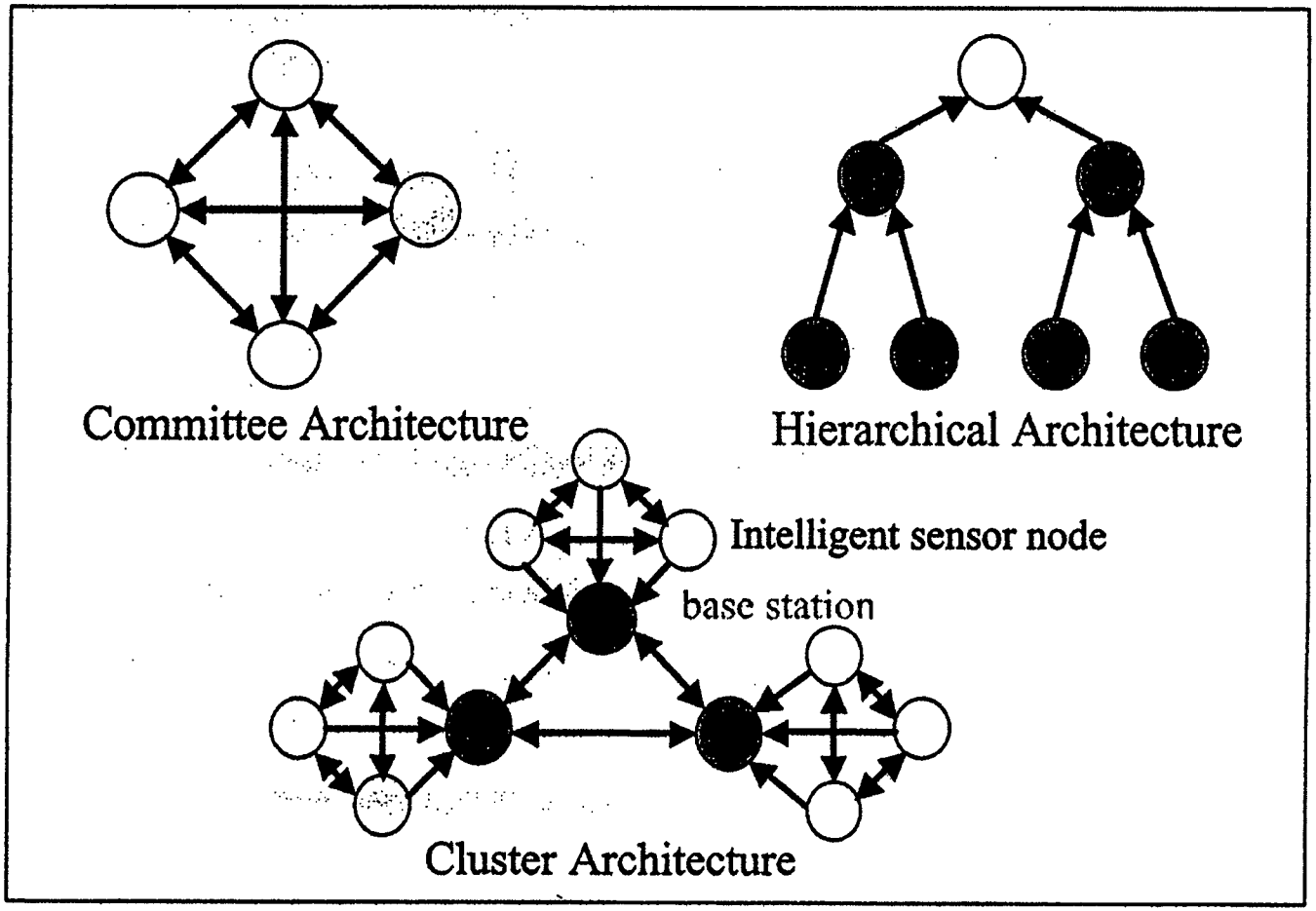
Distributed Micro Flow-Sensor Networks (DMFSN)

- Single sensor system
  - Limited applications and usages
  - Cannot guarantee to deliver accurate information all the time
- DMFSN
  - Built from a collection of intelligent micro flow-sensor nodes
  - Enables more reliable monitoring and control

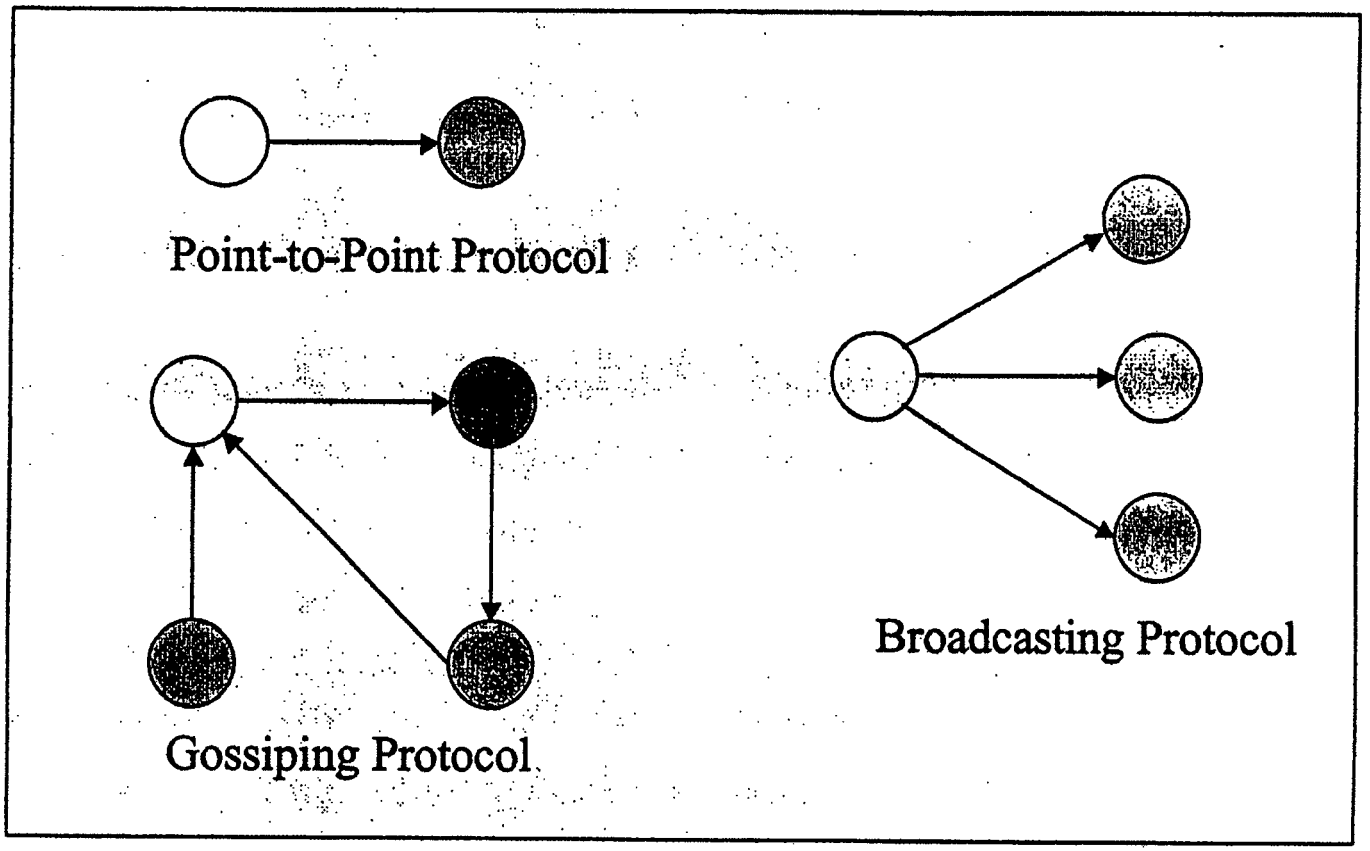
Challenges

- Scaling issues
- Suitable micro-machining technique
- Appropriate sensor network architecture
- Fault-tolerant communication protocol and integration algorithm
- Optimal placement and deployment of micro flow-sensors
- Low communication cost

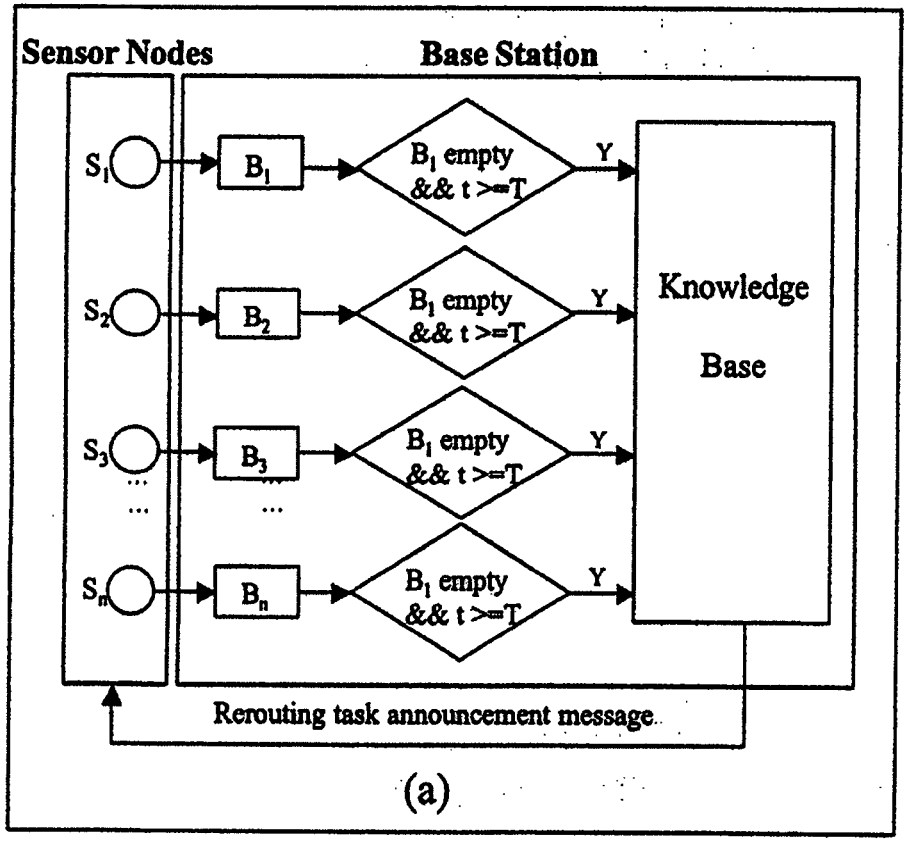
DMFSN Architecture



DMFSN Communication Protocols



DMFSN Communication Protocols (cont.)



(a)

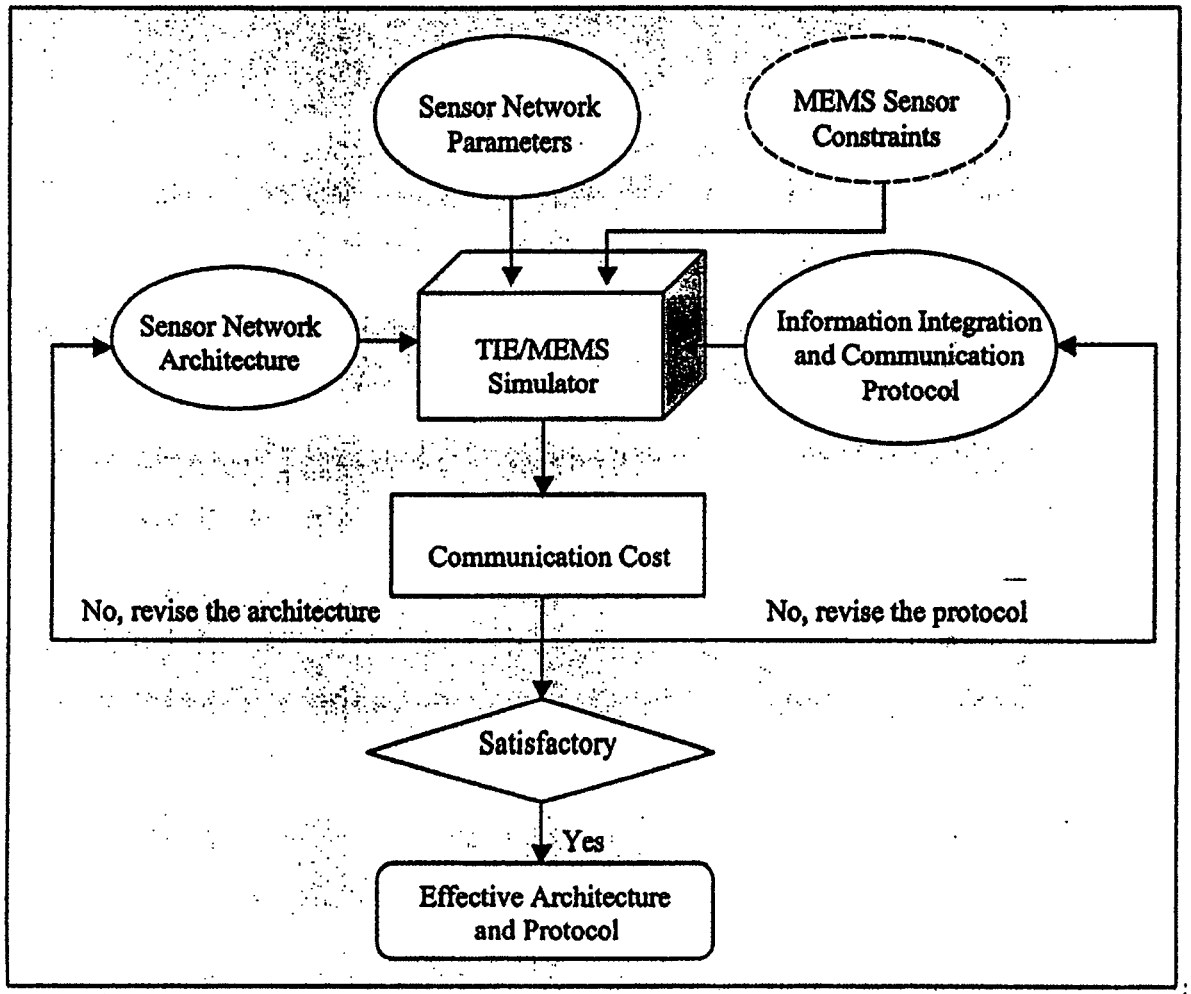
Sensor Node ID	Backup Node	Transmission Delay Time
1	2,3,4	T_1
2	1,4,5	T_2
...	...	...
n	1,2,n-1	T_n

Knowledge Base

(b)

(a) Fault-tolerant Time-out Protocol (FTTP) Model
(b) Knowledge Base Structure

Teamwork Integration Evaluator (TIE)/MEMS



Preliminary simulation results: Normalized communication time vs. No. of sensor nodes *(as ratio of the communication time in the hierarchical architecture)*

#	Sensor Network Alternatives		No. of sensor nodes				
	Architecture	Comm. Protocol	**4** sensor nodes	**8** sensor nodes	**15** sensor nodes	**21** sensor nodes	**30** sensor nodes
1	Committee	PTP	3.1783	1.1027	3.5083	6.7054	N/A*
2	Committee	Broadcasting	1.3852	1.2857	1.8919	1.8335	N/A*
3	Hierarchical	PTP	1	1	1	1	1
4	Cluster	Broadcasting	1.5274	0.2968	0.1145	0.0502	0.0159
5	Cluster	Gossiping	1.1651	0.2811	0.1027	0.0419	0.0145

*Data not available because the number of messages exchanged exceeded the maximum limit supported by Origin 2000

Communication time (cost) vs. No. of sensor nodes

ANOVA Results (α=95%)

No. of Sensor nodes	ANOVA Results
4 sensor nodes	No significant difference among all network architectures and communication protocols
8 sensor nodes	1,2,3 are not significantly different; 4,5 are not significantly different; 1,2,3 are significantly less efficient than 4 and 5
15 sensor nodes	4,5 are not significantly different; 4,5 are significantly different from 1,2,3; 1,2,3 are significantly different; 1>2.>3>4,5
21 sensor Nodes	4,5 are not significantly different; 4,5 are significantly different from 1,2,3; 1,2,3 are significantly different; 1>2>3>4,5
30 sensor nodes	4,5 are not significantly different; 4,5 are significantly more efficient than 3

(1-Committee Arch. + PTP Protocol; 2- Committee Arch.+Broadcasting Protocol;
3-Hierarchical Arch.+ PTP Protocol; 4- Cluster Arch. + Broadcasting Protocol;
5-Cluster Arch.+Gossiping Protocol; i>j means the cost of i is larger than that of j)

Conclusions

- MEMS Flow-Sensors have certain advantages over conventional sensors
- Distributed Micro Flow-Sensor Network (DMFSN) can further enhance the reliability and fault-tolerance of the system
- The best architecture and protocol for the DMFSN depend on the number of micro sensor nodes needed to provide accurate enough information
- The number of sensor nodes required and the optimal deployment of micro sensor nodes depend on the flow environment and sensor physical constraints

The following article entitled "Distributed Micro Flow-Sensor Network (DMFSN) Design and Modeling" provides generalized information about a potential communication protocol.

Distributed Micro Flow-Sensor Network/Array Design and Modeling

Abstract: This paper describes the design issues of the distributed micro flow-sensor networks/arrays (DMFSN/DMFSA) and sensor network architectures and information communication protocols that can be used in DMFSN/DMFSA. A sensor network/array simulator and evaluator, called Teamwork Integration Evaluator (TIE)/MEMS, was developed to evaluate the efficiency of DMFSN in terms of its communication cost. Initial simulation results shows that the cluster sensor network provides better efficiency than hierarchical and committee sensor networks.

Keywords: MEMS, Network Architectures, Protocols, Fault-Tolerant, Time-Out, Teamwork Integration Evaluator

I. INTRODUCTION

The measurement of fluid flows is an important technology used in nearly every industry. Flow sensors are used to measure the movement of the fluid flow (liquid flow or gas flow). They are widely used in a variety of applications, such as automotive applications, environmental monitoring, biomedical applications, etc.

Before the advent of micro minimization, sensor systems tend to be bulky and expensive, so single sensor systems were used to save space and cost. The advantage of single sensor systems is that they are relatively easy to construct and analyze. However, single sensor systems have limited applications and usages. For instance, if we want to measure several variables, say, temperature, pressure and flow density, at the same time, single sensor systems cannot be used. In addition, a single sensor cannot guarantee to deliver accurate information all the time, because it is inevitably challenged by noise or other uncertain disruptions.

Microelectromechanical system (MEMS) technology deals with designing and fabricating entire electrical and mechanical systems, usually on a single silicon chip. Micro flow-sensors are one of the most popular MEMS devices. Due to their small size, the advantages of micro flow-sensors are:

- Interfere less with the environment they are measuring
- Lower manufacturing cost (less materials)
- Can be applied in narrow spaces, such as inside the living organisms, small pipes, automobile engines, etc.
- Multiple sensors can be used with redundancy to improve information accuracy and fault tolerance of the system

Distributed micro flow-sensor networks/arrays (DMFSN/DMFSA) are built from collections of spatially scattered, intelligent micro flow sensor nodes. Each node has the ability to measure the local flow within its accuracy limits, processes the raw sensor data, and cooperates with its neighboring nodes, which can improve the accuracy and reliability of the information. Sensors incorporated with dedicated signal processing functions are called intelligent sensors, or smart sensors. The main roles of dedicated signal processing are to enhance design flexibility and realize new sensing functions. Additional roles are to reduce loads on central processing units and signal transmission lines by distributing information processing to the lower layers of the system (Yamasaki, 1995).

Developing effective DMFSN, however, also faces great challenges, some of which are:

- *Scaling issues.* The differences in dimensions between the micro world and macro world cause differences in balance between magnitudes of forces.
- *Suitable micromachining technique.* To achieve the realization of micro flow sensor structure, the structure has to be released using micromachining techniques (Toshio and Fumihito, 1999).
- *Appropriate network architecture.* Since micro sensor nodes are distributed spatially, appropriate sensor network architecture has to be found to support efficient, undisturbed communication among the sensor nodes.
- *Optimal placement and deployment of micro flow sensors.* Flow measurement is a complex process because of the numerous variables that can affect accuracy, such as flow stability, non-homogeneous flow, temperature variation along the flow path, etc. Hence, the placement and deployment of micro sensors in the flow space (e.g., positioning within the cross-section of a pipe and along the axes of flow) is important in order to minimize the error due to uncertain factors.
- *Fault-tolerant information communication protocol and integration algorithm.* Flow measurement is a complex process, so it is unrealistic to expect all the sensor nodes and communication links along the path to be fault-free all the time.
- *Low communication cost.* Communication cost is often related to communication time and power consumption. Minimizing communication cost is important in online control applications and energy-constrained applications.

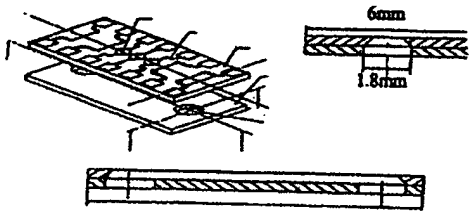
The objective of this paper is to discuss the sensor network architectures and communication protocols that can be used for DMFSN/DMFSA, and the development of a sensor network simulation tool, TIE (Teamwork Integration Evaluator)/MEMS. The purpose of TIE/MEMS is to simulate and evaluate the efficiency of the architectures and protocols. In this paper, some recent proposed micro flow sensor designs are presented in section II; Different types of architectures and communication protocols that can be used for DMFSN are discussed in section III and IV, respectively; Section V describes TIE/MEMS structure and preliminary simulation results. Finally, conclusions and further research are discussed in section VI.

II. MICRO FLOW-SENSOR DESIGN REVIEW

Realization of micro flow-sensors can use several, different operating principles, which include anemometer functions, physical deflections, temperature-dependent layers, time-of-flight, coriolis-force, etc.

Figure 1 illustrates an exploded view of a silicon diaphragm capacitive sensor, called Prandtl micro flow sensor (PMFS), for flow-velocity measurement designed by Berberig *et al.* (1998). Like the well-known classical prandtl tube, the sensor realized flow-velocity detection by measuring the pressure difference between the stagnant fluid pressure in front of the sensor and static pressure in the flow around the sensor. This difference results in a deflection of a silicon diaphragm suspended boss, which servers as the counter electrode of an integrated capacitor that is directly exposed to the fluid to be measured.

Figure 2 shows an example of the mass flow sensor designed by Nguyen and Kiehnscherf (1995). This thermal flow sensor contains polysilicon heaters and measurement resistance on a diaphragm. The fluid flows in an anisotropically etched micro-channel with a cross-section of 0.6mm^2 and a length of 10mm. The channel is covered with a Pyrex glass plate by anodic bonding. A small thermal response time from 1 to 4ms is reached with a diaphragm thickness from 10 to 30um.



1. Polysilicon resistance
2. Aluminum conductor
3. Silicon chip
4. Pyrex plate
5. Flow input and output

Figure 1. Exploded view of the Prandtl micro flow-sensor developed by Berberig *et al.* (1998)

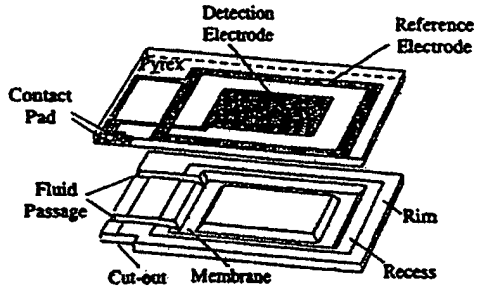


Figure 2. Structure of the thermal flow sensor designed by Nguyen and Kiehnscherf (1995)

III. DISTRIBUTED MICRO SENSOR NETWORK ARCHITECTURE

A distributed micro sensor network is a collection of micro intelligent sensor nodes that are distributed spatially. Its abstraction is a Graph G=(V, E), where V represents the intelligent micro sensor node. This node consists of a microprocessor unit and an associated micro sensor. E represents the communication links connecting the sensor nodes.

The two most common, conventional architectures for distributed sensor networks are the *Committee (or Anarchic)* and the *Hierarchical* architectures, first proposed by Wesson *et al.* (1981). In the Committee structure, each node in the network is autonomous and can send information to all other nodes in the network. Thus if there are N sensors in the network, it needs interconnections that are of the order N^2. The committee architecture is not suitable for a large distributed micro sensor network, because it is too expensive with respect to the communication cost, and not easily extensible (Figure 3(a)). In the Hierarchical structure, sensor nodes are connected to form strict hierarchies. Each node (except the lowest-level leaf node) receives information from the lower level node; integrates the information received according to its position in the hierarchy, and then sends information to the node at the upper level. The root node can be a more powerful processor, or a normal sensor node in which the final result is generated. The Hierarchical architecture has several advantages, such as constant node degree and easy extensibility, but it may not produce accurate information at higher levels because the errors can accumulate when the information goes up the hierarchy. It is also not tolerant to link failures (Figure 3(b)).

Cluster structure overcomes the major limitations of both the Committee and the Hierarchical architectures by limiting the number of communication channels. At the same time, the cluster architecture tolerates the sensor node and link failure. In the cluster architecture, the sensor network is divided into *sensor cluster units (SCU)*, each of which consists of a set of intelligent sensor nodes. An intelligent sensor node consists of a processing unit and an associated micro sensor that measures the variables of interest. A sink node is needed for each cluster to integrate information from all the sensor nodes in the cluster. Considering the complexity of computation and energy constraint of the micro sensor node, a more powerful processor is chosen as the base station for each cluster. Figure 3(c) shows the cluster architecture for the distributed micro sensor network, which is the revision of the network topology proposed by Iyengar *et al.* (1994). In this structure, the base station acts as the control node of the respective SCU, managing information rerouting in case of link failures and combining the information from all the intelligent sensor nodes in the SCU. The intelligent sensor nodes within one SCU (also called sibling nodes) can be fully connected (using broadcasting communication protocol) or connected with only one or several selected sensor node (using gossiping communication protocol), depending on the link failure probability of the system. Broadcasting and gossiping communication protocols are described in more detail in part IV. Base stations in the network can also be fully or partially interconnected, for the same reason as connections among sensor nodes. The external processor is the commander of the distributed micro sensor network, in which information from all the SCUs is integrated to produce a final result, or make decisions as required. The clusters can be deployed along the flow path, depending on the flow situation and applications.

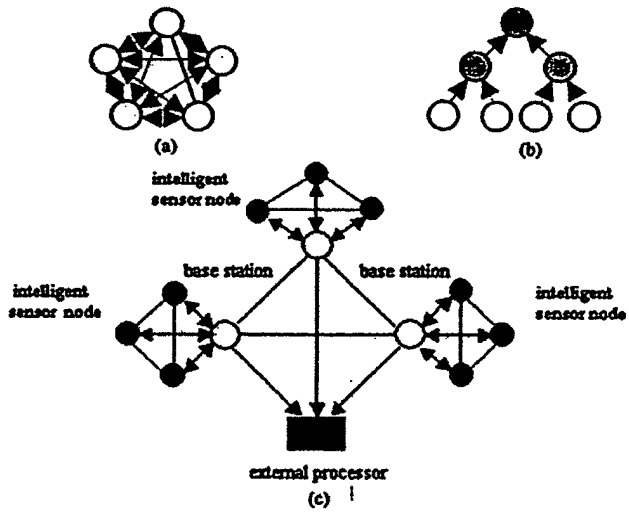


Figure 3. (a) Committee architecture (b) Hierarchical architecture (c) Cluster architecture

IV. INFORMATION COMMUNICATION PROTOCOLS

1. *Point-to-Point communication protocol (PTP).* In this protocol, sensor nodes A and B communicate exclusively with each other without interfering with other nodes. (Ezhilchelvan, 1989, Heinzelman *et al.*, 1999)
2. *Broadcasting communication protocol.* In this protocol, the information sent out by a sensor node is received by all the sensor nodes within a certain range of the sender node. The advantage of this protocol is that when a single node sends information out to a broadcast address, it can reach all its neighboring nodes using only one transmission (Hedetniemi, 1988, Pelc, 1996).
3. *Gossiping communication protocol.* In this protocol, instead of indiscriminately sending information to all its neighboring nodes, each sensor node only forwards the data to one of its neighboring nodes by randomly selecting or according to some criteria, such as the distance between those nodes (Hedetniemi, 1988, Pelc, 1996).
4. *Fault-tolerant time-out protocol (FTTP).* This protocol is an improvement of the traditional time-out protocol with fault-tolerant properties. According to the time-out protocol, the receiver node will stop waiting for the information from the sender node if a certain amount of time, T, has passed. If the committee architecture or cluster architecture is used for the sensor network, the receiver node can announce the rerouting task to the sensor nodes, which have the backup information of the sensor node with the failed link. In order to reduce the message traffic and limit the unnecessary communication, instead of broadcasting the task announcement message to all other nodes, the receiver node can only send the message to those most suitable nodes based on their previous performance. The receiver nodes have to have "knowledge" about the communication ability of those nodes in order to make such "intelligent" decision. Therefore, a database that stores such knowledge needs to be built in the receiver node. In the cluster sensor network, the base stations and the external processor are examples of such receiver nodes. For the committee sensor network, however, it is not economic for each sensor node to have a knowledge base in its dedicated processing unit. Figure 4 (a) shows the fault-tolerant time-out protocol for the cluster sensor network, in which $S_1, S_2,\ldots,S_n$ are the micro sensor nodes, n is the number of sensor nodes in the cluster; $B_1, B_2, B_3,\ldots,B_n$ are the buffers for sensor nodes $S_1, S_2, S_3\ldots,S_n$ respectively. Figure 4(b) shows the knowledge base structure, which consists of three attributes, the sensor node ID, backup nodes which have the redundant information of the sensor node, and the transmission delay time for the sensor node. Transmission delay time is the measure of the performance of the sensor node, which is equal to the difference between the time when the base station receives the

information from the sensor node and the time when the base station issues the task announcement to the sensor node.

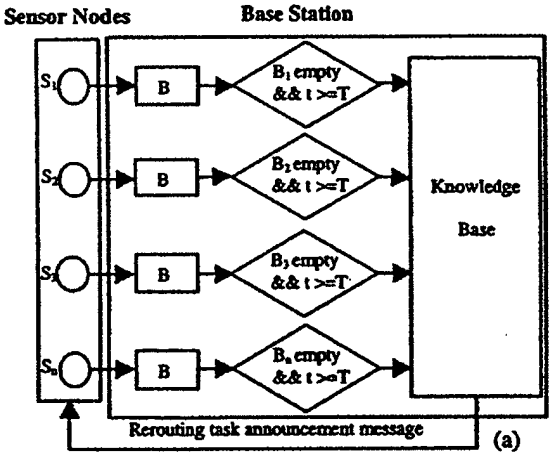


Figure 4. (a) Fault-tolerant time-out protocol (FTTP) model
(b) Knowledge base structure

V. TIE (TEAMWORK INTERGRATION EVALUATOR) / MEMS

The objective of TIE/MEMS, an extended version of the TIE (Teamwork Integration Evaluation) Simulator developed by PRISM (Khanna and Nof, 1994, Huang and Nof, 1996, Anussornnitisam and Nof, 2000), is to simulate and evaluate the architecture and information integration and communication protocol developed for the distributed micro sensor network. TIE/MEMS is written using MPI (Message Passing Interface)(Snir et *al.*, 1996). MPI can be used to simulate the different communication schemes among the sensor nodes, such as point-to-point communication, collective communication, group communication, and so on. In addition, implementations of MPI on top of standard Unix inter-processor communication provide portability to multiple processors, workstation clusters, and heterogeneous networks of workstations (NOW).

5.1. Structure of TIE/MEMS

The overall structure of TIE/MEMS is depicted in Figure 5. Distributed micro sensor network architecture, information communication protocol, sensor network parameters, such as the number of intelligent micro sensor nodes, and constraints unique to MEMS sensors (e.g., physical and chemical constraints causing micro-stress, proximity noise, and other disturbances to measurement and communication) are the inputs of TIE/MEMS; the outputs are the communication costs, which are represented by the simulation time and energy consumption. If the communication costs are too high, we need to revise the network architecture and/or the information communication protocol. Table 1 lists some examples of constraints related to MEMS sensor operations.

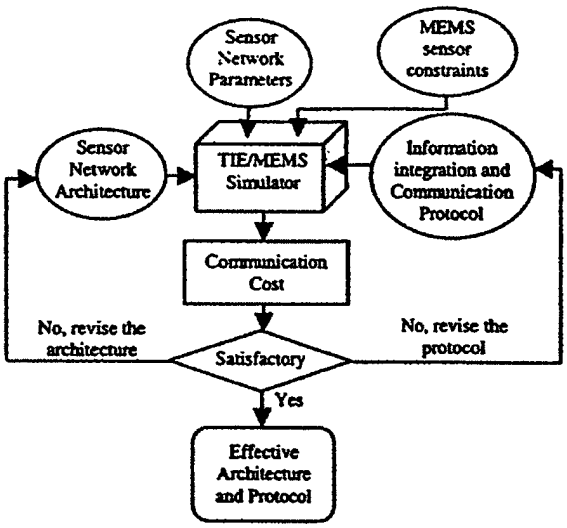


Figure 5. Structure of TIE/MEMS

Table 1. Examples of constraints related to MEMS sensors

Scaling issues	Viscosity force; Surface effect
Problems in micromachining	Squeeze film damping; Particular contamination; Static stiction
Response	Decrease of response time due to protective gels, coatings or diaphragms.
Reliability of microelectronic device	Breakdown of a wire-bond to a junction; Defective encapsulation; Defective semiconductor material; Thermal runaway
Life time	Aging; Contamination of the device; Limited power supply
Noise/interferences	Degraded signal due to the variance of temperature, humidity, etc.

5.2. Preliminary simulation result

Preliminary simulations were run on Origin 2000 workstations at Computer Science in Purdue University. Table 2 shows the results of normalized communication time from 90 simulation runs for the sensor network architectures and communication protocols described in section IV. The communication time of the hierarchical network is used as the baseline. Table 3 shows the ANOVA results (α=0.95) of the data in Table 2. From Table 3, it can be seen that when the number of sensor nodes is small, the communication cost of all network architectures and communication protocols is similar. Therefore, committee architecture or cluster architecture might be preferred in order to improve the network fault-tolerance. When the number of sensor nodes increases, the performance of committee network, especially the committee network with point-to-point communication protocol deteriorates dramatically, so the hierarchical network and cluster network may be preferred. In either case, cluster network is a recommended choice.

Two important points need to be mentioned:

1) The communication cost of the hierarchical sensor network is greatly influenced by the height and fan-out of the tree structure. The less height and larger fan-out the structure has, the smaller the communication cost is, but the more vulnerable it is to the link and/or node failures.

2) The communication cost of the cluster sensor network is greatly influenced by the number of sensor nodes in each cluster and the number of clusters. The less the number of sensor nodes in each cluster, and the more clusters the sensor network has, the less communication time needed within each cluster, but more communication time may be needed among the clusters, so we need to calculate the optimal number of clusters and number of sensor nodes in each cluster to minimize the communication cost.

Table2. TIE/MEMS analysis results: Normalized communication time vs. No. of sensor nodes (as ratio of the communication time in the Hierarchical architecture)

#	Sensor network alternatives		No. of sensor nodes				
	Architecture	Communication Protocol	4 sensor nodes	8 sensor nodes	15 sensor nodes	21 sensor nodes	30 sensor nodes
1	Committee	PTP	3.1783	1.1027	3.5083	6.7054	N/A*
2	Committee	Broadcasting	1.3852	1.2857	1.8919	1.8335	N/A*
3	Hierarchical	PTP	1	1	1	1	1
4	Cluster	Broadcasting	1.5274	0.2968	0.1145	0.0502	0.0159
5	Cluster	Gossiping	1.1651	0.2811	0.1027	0.0419	0.0145

*Data not available because the number of messages exchanged exceeded the maximum limit supported by Origin 2000

Table3. AVONA results (α=95%) of communication time (cost) vs. No of sensor nodes (1,2,3,4,5 correspond to # in Table 2 respectively; i>j means the cost of i is larger than that of j)

No. of sensor nodes	ANOVA results
4 sensor nodes	no significantly different among all network architectures and communication protocols
8 sensor nodes	1, 2 and 3 are not significantly different, 4 and 5 are not significantly different, but 1,2,3 are significantly different from 4 and 5 and 1,2,3>4,5
15 sensor nodes	4,5 are not significantly different, but 4,5 are significantly different from 1,2,3 and 1,2,3 are also significantly different. 1>2>3>4,5
21 sensor nodes	4,5 are not significantly different, but 4,5 are significantly different from 1,2,3 and 1,2,3 are also significantly different. 1>2>3>4,5
30 sensor nodes	4,5 are not significantly different, but 4,5 are significantly different from 3 and 3>4,5

VI. CONCLUSIONS AND FUTURE RESEARCH

Alternative sensor network architectures and communication protocols are discussed and compared in the paper. TIE/MEMS is a modeling tool developed to simulate and evaluate the cost effectiveness of the network architectures and communication protocols.

Several other issues, such as the optimal deployment of micro flow sensors, and fault tolerant algorithms, are also considered in this research. These issues depend on the flow environment, sensor physical constraints like data accuracy, time-dependent behavior, and energy constraints (for wireless communication). Therefore, in the next step, those constraints will be investigated to find out the number of sensor nodes needed to achieve the required data accuracy. TIE/MEMS will be applied to find the recommended sensor architecture and communication protocols to minimize the communication cost and yield the best sensory information.

REFERENCE

1. Anussornnitisam, P. and Nof, S.Y., "a Teamwork Integration Evaluator for coordination protocols", *Proceedings of ICPR-2000*, Aug. 2000
2. Berberig, O., Nottmeyer, K., Mizuno, J., Kanai, Y. and Kobayashi, T., "The Prandtl micro flow sensor (PMFS): a novel silicon diaphragm capacitive sensor for flow-velocity measurement", *Sensors & Actuators*, v A66, n 1-3, 1 Apr. 1998, p. 93-98
3. Ezhilchelvan, P.D., Shrivastava, S.K. and Tully, A., "Constructing replicated systems using processors with point to point communication links", *IEEE Computation Society Press*, 1989, p. 177-184
4. Hedetniemi, S.M., Hedetniemi, S.T. and Liestman, A.L., "A survey of gossiping and broadcasting in communication networks", *Networks*, v 18, n 4, Winter 1988, p. 319-349
5. Heinzelman,W., Kulik, J. and Balakrishnan, H., "Adaptive protocols for information dissemination in wireless sensor networks", *Proceedings of the Annual International Conference on Mobile Computing & Networking*, 1999, p. 174-185
6. Huang, C.Y. and Nof, S.Y., "TIE: Teamwork Integration Evaluation Simulator: A manual for TIE: 1.2", *Research Memorandum*, No 96-2, School of Industrial Engineering, Purdue University, 1996
7. Iyengar, S.S., Jayasimha, D.N. and Nadig, D., "A versatile architecture for distributed sensor integration problem", *IEEE Transaction on Computers*, v 43, n 2, Feb. 1994, p. 175-185
8. Khanna, N. and Nof, S.Y., "TIE: Teamwork Integration Evaluation Simulator: A preliminary User Manual for TIE 1.1", *Research Memorandum*, No. 94-21, School of Industrial Engineering, Purdue University, 1994
9. Nguyen, N.T. and Kiehnscherf, R., "Low-cost silicon sensors for mass flow measurement of liquids and gases", *Sensors & Actuators*, v A49, n 1-2, Jun. 1995, p. 17-20
10. Pelc, A., "Fault-tolerant broadcasting and gossiping in communication networks", *Networks*, v 28, n 3, Oct. 1996, p. 143-156
11. Rasmussen, A. and Zaghloul, M.E., "In the flow with MEMS", *IEEE Circuits & Devices Magazine*, v 14, n 4, July 1998, p. 12-25
12. Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W. and Dongarra, J., "MPI: The complete reference", MIT press, 1996
13. Toshio, F. and Fumihito, A., "Microrobotics", *Handbook of Industrial Robotics*, Second Ed., 1999
14. Wesson, R., Hayes-Roth, F., Burge, W.J., Stasz, C. and Sunshine, C.A., "Network Structures for Distributed Situation Assessment", *IEEE Transaction on System*, 1981, p. 5-23
15. Yamasaki, H., "What are the intelligent sensors?" *Handbook of Sensors and Actuators 3, Intelligent Sensors*, Yokogawa Research Institute Corporation, Tokyo, Japan, 1995

The following materials entitled "DISTRIBUTED MICRO FLOW-SENSOR ARRAYS AND NETWORKS: DESIGN OF ARCHITECTURE AND FAULT-TOLERANT INTEGRATION" provide a description of various embodiments of the present invention.

DISTRIBUTED MICRO FLOW-SENSOR ARRAYS AND NETWORKS:

DESIGN OF

ARCHITECTURE AND FAULT-TOLERANT INTEGRATION

TABLE OF CONTENTS

Page of this paper

Page of this paper

LIST OF TABLES

Table Page of this paper

Table

Page of this paper

LIST OF FIGURES

Figure — Page of this Paper

Figure Page

NOMENCLATURE

Variable		Page of this paper
m_{ki}	Number of Tasks offered to b_{ki}	18
N_b	Number of Backup Nodes	17
N_c	Number of Sensor Nodes in the Cluster	22
N_l	Number of Failed Links	23
N_s	Number of Sensor Nodes	9
NA	Network Architecture	3
r_{bs}	Ratio of c_l of FTTP with Blind Base Station and FTTP with Smart Base Station	29
r_d	Deterioration Rate	46
r_l	Link Failure Rate	17
S	Connected Interval Sequence	32
t_{ki}	Time When b_{ki} finishes the Task	20
T	Time-out Period	18
u_i	Upper Bound of Interval i	32
U_i	Upper Bound of *AS i*	31
V	The Set of Sensor Nodes	8
v_i	Sensor Node *i*	8
$\Delta\tau_i$	Errors of Readings from τ^*	35
ε_i	Maximum Tolerable Variations of Readings of v_i	31
γ_{ki}	Transmission Rate of b_{ki}	18
τ_i	Readings of v_i	31
τ^*	True Value of the Variable of Interest	35

Acronym		
AS	Abstract Sensor	31
BDM	Required Data Message	19

Acronym		Page of this paper
BP	Broadcasting Protocol	11
CLA	Cluster Architecture	16
COA	Committee Architecture	9
CS	Concrete Sensor	31
DMFSA/DMFSN	Distributed Micro Flow-Sensor Arrays and Networks	2
EWGP	Evenly Wide Gossiping Protocol	17
FTA	Flat Tree Architecture	9
FTSIA	Fault-tolerant Sensor Integration Algorithm	4
FTTP	Fault-tolerant Time-out Protocol	18
GP	Gossiping Protocol	11
HIA	Hierarchical Architecture	18
MBDA	Multilevel binary de Bruijn Architecture	9
MLN	Multilevel Network	9
PTP	Point-to-Point Protocol	11
RTM	Rerouting Task Message	18
SCU	Sensor Cluster Unit	16
TIE	Teamwork Integration Evaluator	3
TOM	Task Offer Message	19

ABSTRACT

The measurement of fluid flow is an important technology used in nearly every industry. Flow-sensors measure the movement of fluid flow (liquid or gas flow). Conventionally, single sensor systems are used to save space and cost. The development of MEMS enables production of large amounts of micro sensors at low cost. Distributed micro flow-sensor arrays and networks (DMFSA/DMFSN) are built from collections of spatially scattered, intelligent micro flow-sensor nodes. Each node is able to measure the local flow within its accuracy limits, process the raw sensor data, and cooperate with its neighboring nodes. Multi-sensor architectures can improve the reliability and fault-tolerance of the system. The objective of this research is to address the design of the network architectures, communication protocols, and sensor information integration algorithm, for DMFSA/DMFSN, by developing a MEMS sensor network simulation tool, called TIE (Teamwork Integration Evaluator)/MEMS to analyze the efficiency of the sensor network architectures and communication protocols. New cluster network architecture, two versions of fault-tolerant time-out communication protocol (FTTP) – one with blind base stations, and one with smart base stations, and a fault-tolerant sensor integration algorithm (FTSIA) are proposed and analyzed in this thesis. TIE/MEMS is applied to evaluate five combinations of network architectures and communication protocols and compare the two versions of FTTP. The experiment results show that under certain conditions the proposed cluster architecture can support efficient information communication and integration in DMFSA/DMFSN because it requires relatively low

communication cost and tolerates possible sensor and link failures. The results also show that although the FTTP with smart base station requires an additional knowledge base in its structure compared with the one with blind station, it significantly reduces the communication time. In addition, in order to evaluate the effectiveness of the proposed FTSIA, three cases of simulation experiments were conducted to compare the output results from FTSIA and statistical calculations. The experiment results indicate that the FTSIA yields more accurate measurement results than non-FTSIA statistical calculation, when some of the sensors yield faulty information. An empirical experiment with an array of eight pressure sensors was developed to illustrate FTSIA. General design guidelines are developed based on the experiments.

CHAPTER 1 INTRODUCTION TO THE PROBLEM

The measurement of fluid flow is an important technology used in nearly every industry. Flow-sensors are used to measure the movement of fluid flow (liquid flow or gas flow). They are widely used in a variety of applications, such as automotive applications, environmental monitoring, biomedical applications, etc.

1.1. Disadvantages of Single Sensor Systems

Before the advent of micro minimization, sensor systems tended to be bulky and expensive, so single sensor systems were used to save space and cost. Although single sensor systems are relatively easy to construct and analyze, they have at least two disadvantages:

1. They have limited applications and usages. For instance, if several variables, say, temperature, pressure and flow density, need to be measured at the same time, single sensor systems cannot be used.
2. In addition, a single sensor cannot guarantee delivery of accurate information all the time, because it is inevitably affected by noise or other uncertain disruptions.

Thus, single sensor systems are not suitable for critical applications.

1.2. MEMS Flow-Sensors in Distributed Arrays and Networks

Microelectromechanical system (MEMS) technology deals with design and fabrication of entire electrical and mechanical systems, usually on a single silicon chip. The development of MEMS enables production of large amounts of micro sensors at low cost. Micro flow-sensors are one of the most common MEMS devices.

Due to their small sizes, the advantages of micro flow-sensors are:

1. Interfere less with the environment they are measuring
2. Lower manufacturing cost
3. Can be applied in narrow spaces, such as inside living organisms, small pipes, automobile engines, etc.
4. Multiple sensors can be used with redundancy to improve information accuracy and fault tolerance of the system.

Distributed micro flow-sensor arrays and networks (DMFSA/DMFSN) are built from collections of spatially scattered, intelligent micro flow-sensor nodes. Each node has the ability to measure the local flow within its accuracy limits, process the raw sensor data, and cooperate with its neighboring nodes. Sensors incorporated with dedicated signal processing functions are called *intelligent sensors*, or *smart sensors*. The main roles of dedicated signal processing units are to enhance design flexibility and realize new sensing functions. Additional roles are to reduce loads on central processing units and signal transmission lines by distributing information processing to the lower layers of the system (Yamasaki, 1995). In this thesis, the micro flow-sensor array refers to a group of the same type of micro flow-sensors which are placed close together to measure the same variable of interest. Different groups of micro flow-sensor arrays (either of the same type or different types) can be distributed widely in the flow environment to form a micro flow-sensor network.

1.3. Problem Definition

Developing effective DMFSA/DMFSN faces great challenges. Five major design issues are:

1. Suitable micromachining technique. To achieve the realization of micro flow-sensor structure, the structure has to be released using suitable micromachining techniques (Toshio and Fumihito, 1999).

2. Appropriate network architecture, *NA*. Since micro sensor nodes are distributed spatially, appropriate sensor network architecture has to be found to support efficient communication among the sensor nodes.
3. Cost-efficient communication protocol, *CP*. Communication cost C is often related to communication time, c_t, and energy consumption, c_e, of the system. Minimizing communication cost is important in the online control and energy-constraint applications.
4. Fault-tolerant communication protocol and information integration algorithm, *IA*. It is unrealistic to expect all the micro sensor nodes and communication links along the path to be fault-free all the time because of the constraints of micro sensors and complex flow environment.
5. Optimal placement and deployment of micro flow-sensors. Flow measurement is a complex process because of the numerous factors that can affect accuracy, such as flow stability, non-homogeneous flow, etc. Hence, the placement and deployment of micro sensors in the flow space is important in order to minimize the error due to the uncertain factors.

1.4. Scope and Objectives

The objective of this research is to address the issues No. 2, 3 and 4 in section 1.3 in the design of the network architectures, *NA*, communication protocols, *CP*, and sensor information integration algorithm, *IA*, for DMFSA/DMFSN, Issues No. 1 and 5 are out of the scope of this thesis.

In Chapter 2, some recent research is reviewed on the design of micro flow-sensors, sensor network architectures, communication protocols, and multi-sensor fusion and integration. The proposed cluster architecture and fault-tolerant time-out communication protocol (FTTP) for DMFSA/DMFSN are presented in Chapter 3. In order to evaluate the sensor network architectures and protocols developed for DMFSA/DMFSN, TIE(Teamwork Integration Evaluator) /MEMS is developed. In Chapter 4, the background and structure of TIE/MEMS are described. In addition, two sets of simulation

experiments were conducted to illustrate the application of TIE/MEMS: one is to evaluate the efficiency of five combinations of sensor network architectures and communication protocols based on the communication time and energy consumption; and the other is to evaluate and compare two alternatives of FTTP, FTTP with blind base station and with smart base station. Those experiments were conducted on SGI Origin 2000 workstation at the Department of Computer Science in Purdue University. In Chapter 5, the proposed fault-tolerant sensor integration algorithm (FTSIA) is presented. Three cases of simulation experiments were conducted to test the effectiveness of this algorithm. In these experiments, the results from FTSIA were compared with the statistical mean of the same simulated measurement values. In chapter 6, a case study of measuring the airflow using a pressure sensor array is described to illustrate the application of FTSIA in real flow measurement. The experiment setup and results are also described. Finally, the design recommendations of DMFSA/DMFSN are summarized, and the conclusions and future research are discussed in Chapter 7.

1.5. Contributions and Significance of the Research

This research is being undertaken in response to the real problems encountered in the flow measurement area. Given the widespread use and importance of flow-sensors, the limitations of the current flow measurement technology are surprising. DMFSA/DMFSN opens original new directions for this area. Some of the main contributions of this research are:

1. Intelligent Sensors manufactured using advanced IC (Integrated Circuit) technology will fundamentally change the nature of sensing and control systems.
2. The ability to place the micro flow-sensors when and where they are needed enables better control of the process.
3. Redundancy enables more reliable monitoring and control because occurrence of failures of some sensors will not inhibit the sensing ability of the system.

CHAPTER 2 LITERATURE REVIEW

This Chapter reviews the recent research on the design of micro flow-sensors, sensor network architectures, communication protocols, and multi-sensor fusion and integration.

2.1. Micro Flow-Sensor Design

Realization of micro flow-sensors can rely on several, different operating principles. The most common ones include:

1. Anemometer

A thermal anemometer refers to sensors that measure total heat loss. The heat loss is due to some heating element (such as a polysilicon resistor) heating the fluid. The heat loss depends on the flow rate of the fluid and increases or decreases with the increase of flow, depending on the operating mode for the structure (Rasmussen and Zaghloul, 1998). Anemometer is the most commonly used type of micro flow-sensor (For instance, Nguyen and Kiehnscherf, 1995; Mayer et al., 1997.) Figure 2.1 shows the mass flow-sensor designed by Nguyen and Kiehnscherf (1995). This thermal flow-sensor contains polysilicon heaters and measurement resistance on a diaphragm. The fluid flows in an anisotropically etched micro-channel with a cross-section of $0.6mm^2$ and a length of 10mm. The channel is covered with a Pyrex glass plate by anodic bonding. A small thermal response time from 1 to 4ms is reached with a diaphragm thickness from 10 to 30μm.

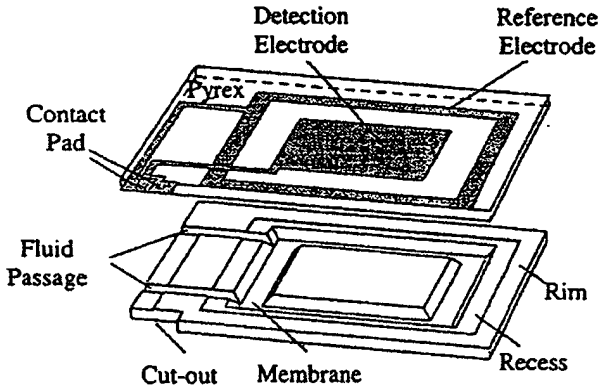


Figure 2.1 Structure of the Mass Flow-sensor Designed by Nguyen and Kiehnscherf (1995)

2. Time-of-Flight

A good method to measure flow would be by marking the flowing medium and detecting the movement of the mark. For example, a mark can be a floating object which can move with the medium while being stationary with respect to the medium. The time that it would take the object to move with the flow from one position to another could be used for the calculation of the flow rate (For example, Shofi and Esashi, 1994; Van Kuijk et al., 1994.)

Both anemometer and sensors using the thermal time-of-flight principle belong to thermal transport flow-sensors. They are more sensitive than other types of flow-sensors and have a broad dynamic range. They can be employed to measure very minute gas or liquid displacements, as well as fast and strong currents. Major advantages of these sensors are the absence of moving components and the ability to measure very low flow rates.

3. Pressure Gradient

A fundamental equation in fluid mechanics is the *Bernoulli Equation*, which is strictly applicable only to the steady flow of nonviscous, incompressible medium. With the Bernoulli Equation the fluid velocity can be found by measuring pressures along the flow. The pressure gradient flow measurement requires introducing flow resistance. The concept is analogous to Ohm's law: voltage (pressure) across a fixed resistor is proportional to current (flow). The advantages of the pressure gradient method include the absence of moving components and use of the standard pressure sensors which are

readily available. One of its disadvantages, however, is the restriction of flow by resistive devices. An example of the pressure flow-sensor is shown in Berberig et al. (1998).

4. Ultrasonic Sensors

Flow can be measured by employing ultrasonic waves. The main idea behind this principle is the detection of frequency or phase shift caused by the flowing medium. One possible implementation is based on the *Doppler Effect*. According to the Doppler Effect, when the target moves toward or away from the transmitting antenna, the frequency of the reflected radiation will change. An alternative method relies on the detection of the increase or decrease of the effective ultrasound velocity in the medium. The effective velocity of sound in a moving medium is equal to the velocity of sound relative to the medium plus the velocity of the medium with respect to the source of the sound. The difference between the velocity of the sound wave propagating upstream and the one propagating downstream is exactly twice the velocity of the medium, which allows one to determine the velocity of the flow. Several researchers have implemented the ultrasonic flow-sensors (For example, von Jena and Magori, 1995; Ishikawa et al., 2000.) An advantage of the ultrasonic flow-sensor is its ability to measure flow without direct interference with the fluid.

5. Physical Deflection

One example of a micro flow-sensor using the physical deflection principle is the device designed by Gass et al. (1993). It measures the drag force exerted by the fluid onto an obstacle – a cantilever beam having overall dimensions of 0.03mm × 1mm × 3mm. This sensor can detect very low flow (down to nanoliter resolution).

2.2. Sensor Network Architecture

The abstraction of a sensor network can be represented as $NA = (V, E)$, where V is the set of sensor nodes and E is the set of communication links connecting the sensor nodes. $V = \{v_i\}(i = 1,2,..., N_s)$, where v_i is the sensor node i and N_s is the number of sensor nodes

in the network. $E = \{e_{ij}\}(i = 1,2,...,N_s; j = 1,2,...,N_s)$, where e_{ij} is the communication link between v_i and v_j.

Two most conventional network architectures that can be used for sensor networks are the *committee architecture* (COA) and the *hierarchical architecture* (HIA), first discussed by Wesson et al. (1981). In the COA (Figure 2.2(a)), each node in the network is autonomous and can broadcast information to all other nodes in the network. Thus, if N_s is equal to n, it needs interconnections that are of the order $O(n^2)$. With such a high number of interconnections, this architecture is robust relative to link failures because the messages from a source node to a destination node can be easily rerouted by bypassing the failed links. The COA, however, is not suitable for a sensor network with a large value of N_s, because it is too expensive with respect to communication cost, C, and not easily extensible. In the HIA, nodes are connected to form strict hierarchies. Each node (except the lowest-level leaf nodes) receives information from the next lower level node, integrates the information received according to its position in the hierarchy, and then sends information to the node at the next upper level. The root node is the node at the highest level, which is responsible for interpretation of data at the global level and for decision-making. For a network with N_s equal to n, HIA requires $O(n)$ interconnections. A special case of HIA is a complete binary tree (Figure 2.2(b)). HIA has several advantages, including constant node degree and easy extensibility, but it may not produce accurate information at higher levels because the errors can accumulate when the information is transmitted up the hierarchy. In addition, HIA is not tolerant to link failures.

Two other sensor network architectures, proposed by Iyengar et al. (1990) and Iyengar et al. (1994), are the *Flat Tree architecture* (FTA) and *Multilevel binary de Bruijn architecture* (MBDA). In FTA, the network is partitioned into clusters and each cluster is internally organized as a binary tree. There are three types of nodes in each cluster: leaf nodes, intermediate nodes and root nodes. Leaf nodes are at the first level of the network and the parent of a node is at one level higher than its children. All the root nodes in the network are fully interconntected (Figure 2.2(c)). FTA is not robust to link failures. MBDA is a *multilevel network* (MLN) with the top level fully connected and with each of the other levels interconnected as a de Bruijn network. MLN is a network in which each

node of the network can be associated with a level number. Assuming that a node i is at level m, then the neighbors of node i are the set of nodes at the same level, that is m, to which node i is connected. The parents of node i are the set of nodes at level m-1 to which node i is connected. The children of node i are the set of nodes at level m+1 to which i is connected. If each node in the network can have at most one parent and r children, the network is called *r-ary multilevel network*. Definitions and more details about the de Bruijn graph and network can be found in Leighton (1992). MBDA is not suitable for the DMFSA/DMFSN because of its complex structure.

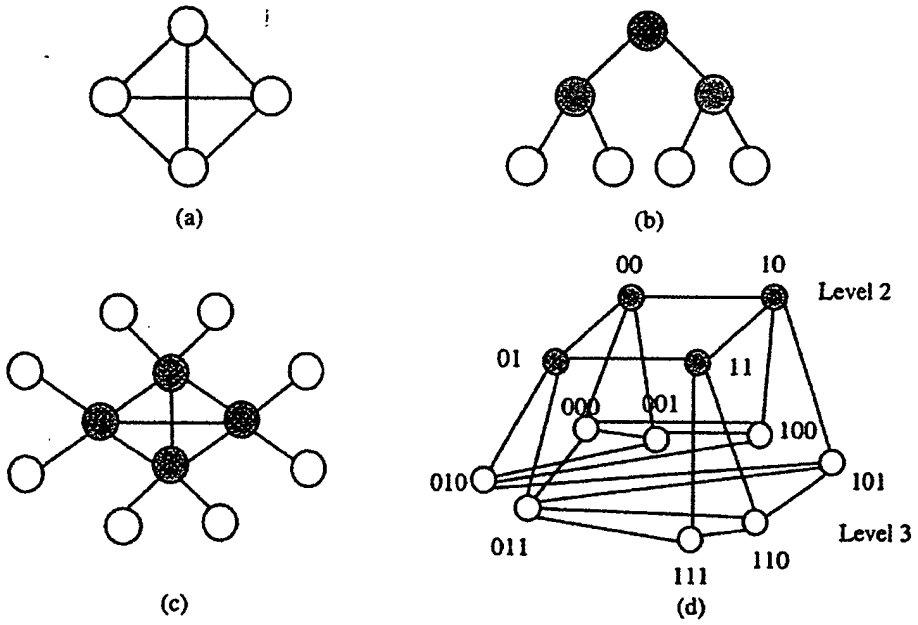


Figure 2.2 Network Architectures

(a) Committee Architecture, COA (b) Hierarchical Architecture, HIA
(c) Flat Tree Architecture, FTA (d) Multilevel Binary de Bruijn Architecture, MBDA

2.3. Communication Protocol

In recent years, there has been increasing interest in the development of communication protocols for the distributed micro sensor network, especially the wireless micro sensor network (for example, Utete and Durrant-Whyte, 1994; Gossink et al., 1998; Agre et al.,

1999; Chandrakasan et al., 1999; Heinzelman et al., 1999; Bhushan and Rengaswamy, 2000.) The sensor network can improve the accuracy of information obtained via collaboration among sensor nodes.

The simplest communication protocol is the *point-to-point protocol* (PTP). In this protocol, sensor nodes A and B communicate exclusively with each other without interfering with other nodes (Figure 2.3(a)).

Another commonly used communication protocol is the *broadcasting protocol* (BP). In this protocol, the information sent out by the sender node is received by all the other nodes within a certain range of the sender node (Figure 2.3(b)). The advantage of this protocol is that when a single node sends information out to a broadcast address, it can reach all its neighboring nodes using a single transmission. BP is, therefore, $\frac{1}{n}$ cheaper than PTP protocol in one-to-many communication, where n is the number of neighboring nodes for the sender node. BP is a common type of the *classic flooding protocol*, which means that upon receiving a piece of data, each node then stores and sends a copy of the data to all of its neighbors. It disseminates data quickly in a network where bandwidth is not scarce and links are not loss-prone. However, since a node always sends data to its neighbors, regardless of whether or not the neighbor has already received the data from another source, it leads to the implosion problem and wastes resources by sending duplicated copies of data to the same node.

Gossiping protocol (GP) is an alternative to the classic flooding protocol, in which instead of indiscriminately sending information to all its neighboring nodes, each sensor node only forwards the data to one randomly selected neighbor (Heinzelman *et al.*, 1999) (see Figure 2.3(c)). While the GP distributes information more slowly than BP, it dissipates resources, such as energy, at a lower rate as well. In addition, it is not as robust relative to link failures as BP because a node can only rely on one other node to re-send the information for it, in case a link failure happens.

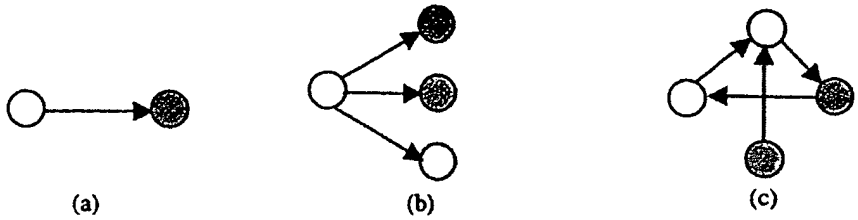


Figure 2.3 Communication Protocols
(a) Point to Point Protocol, PTP (b) Broadcasting Protocol, BP
(c) Gossiping Protocol, GP

In order to solve the problem of implosion, a sensor protocol for Information via Negotiation (SPIN) was proposed by Heinzelman et al. (1999). SPIN nodes negotiate with each other before transmitting data, which helps ensure that only useful information will be transferred. SPIN nodes use three types of messages to communicate: ADV (new data advertisement), REQ (request for data) and DATA (data message). SPIN protocol works in three stages (ADV-REQ-DATA). The protocol starts when a node advertises the new data that is ready to be disseminated. It advertises by sending an ADV message to its neighbors, naming the new data (ADV stage). Upon receiving an ADV, the neighboring node checks to see whether it has already received or requested the advertised data. If not, it responds by sending an REQ message for the missing data back to the sender (REQ stage). The protocol completes when the initiator of the protocol responds to the REQ with a DATA message, containing the missing data (DATA stage). Figure 2.4 shows an example of this protocol.

In a relatively large sensor network, there is typically no central base station to which all nodes can directly communicate, and thus a low-power routing protocol is needed to route the data from the sensors to the high-powered base-station. An energy efficient protocol for the networked sensors was proposed in Chandrakasan et al. (1999). According to this protocol, each node maintains a routing table where the cost of each path is the power dissipated in transmitting information along that path. Therefore, each node can transmit data to the high-powered node using a minimum amount of total energy. Furthermore, most nodes in the sensor network "go to sleep" except for a few "monitor-site" sensors that are chosen randomly and reassigned periodically. These monitor sensors run simple

algorithms to detect the presence of an "event". When an event is detected, each monitor-site sensor "wakes up" its neighbors, which in turn "wake up' their neighbors.

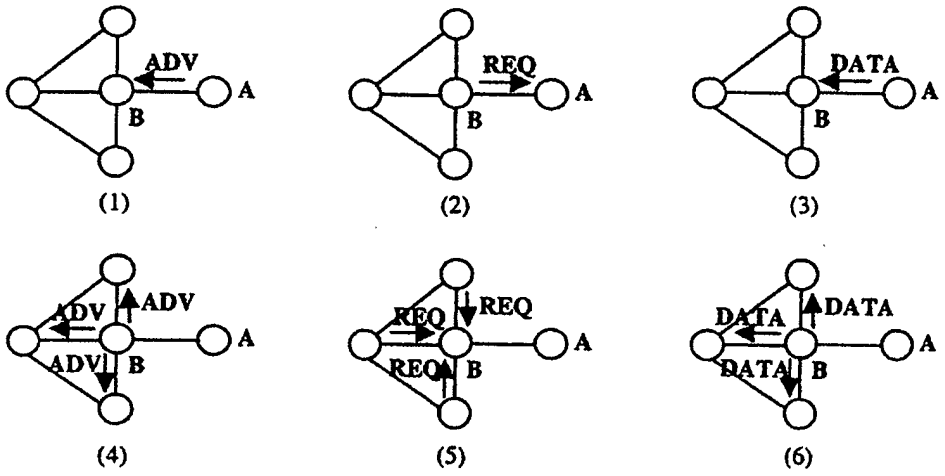


Figure 2.4 An Example of the SPIN Protocol (Heinzelman et al., 1999) (1) node A starts by advertising its data to node B (2) node B responds by sending a request to node A; (3) node A sends data to node B; (4) upon receiving the data, node B sends out advertisement to its neighbors; (5) node B receives the requests from its neighbors and sends data to them

2.4. Multi-Sensor Fusion and Integration

A sensor system must interact with its environment. Measurement errors (such as noise) and sensor failures of single devices are unavoidable. Multi-sensor Fusion and Integration is the process of combining information from different sensors. Using readings from several independent sensors reduces the vulnerability of a system to the failure of a single component. Combining readings from several different kinds of sensors in a system can result in more accurate information than otherwise possible. Possible applications of multi-sensor fusion and integration include virtually all systems involving signal processing, such as in aeronautics (Brown and Olson, 2000), manufacturing (Kuo, 1998), remote sensing (Paul, 2001), medical application (Kubler et al., 2000), etc.

There are several types of multi-sensor fusion and integration depending on the types of the sensors and their deployment (Iyengar et al., 1995):

1. Competitive Integration: Each sensor ideally measures identical information, but in reality, is subject to noise and other disruptions. In this type of integration, even if some sensors are faulty, the information from the erroneous sensors can be discarded during the integration process if a good integration algorithm is used.
2. Complementary Integration: Partial and overlapping (complementary) information is available from each sensor. The partial information must be integrated into complete information. If some sensors are faulty, the result of the integration can be erroneous.
3. Cooperative Integration: Partial and non-overlapping (cooperative) information is available from each sensor. The partial information must be integrated into complete information. No sensor can be faulty; otherwise, the results would be faulty.
4. Independent Integration: Unrelated information is available from the sensors. Integrating them simply means adding more fields to a data record.
5. Temporal Integration: It requires integration of information over a given time. The data, sampled at time intervals, must be processed to re-create the notion of time sequence.
6. Spatial Integration: It requires integration of information over a given space.

Different mathematical tools have been used in the research on multi-sensor fusion and integration. The most commonly used are (Brooks and Iyengar, 1998):

1. Algorithm: An algorithm may be regarded as a finite sequence of instructions to accomplish a given task. In addition to usually having input and output, an algorithm has three properties: each instruction is definite, i.e., its meaning is unambiguous; each instruction is effective, i.e., it can be easily performed by a machine or a person; the sequence of instructions is finite, i.e., it terminates after a finite number of steps.

2. Linear Algebra: It is the branch of mathematics concerned with solving simultaneous linear equations. It provides tools for grouping together and solving sets of related equations.
3. Probability: It is a methodology that has been developed for representing and quantitatively measuring random events. It is the source of most basic methods for dealing with uncertainty.
4. Meta-Heuristics: It is a term used to characterize a number of methods that have been found to be practical algorithms (meaning, not guaranteeing optimality) for solving non-linear problems. It includes tabu search, genetic algorithms, simulated annealing, and artificial neural networks.

In this thesis, it is assumed that all sensors are of the same type and measure the same variable of interest, therefore, competitive integration is the research interest here. A sensor integration algorithm is developed for DMFSA/DMFSN, because it is easily implemented in the processing unit with limited computation power.

CHAPTER 3 DMFSA/DMFSN ARCHITECTURE AND COMMUNICATION PROTOCOL

In this chapter, the proposed cluster network architecture and fault-tolerant time-out communication protocol for DMFSA/DMFSN are presented.

3.1. Cluster Network Architecture for DMFSA/DMFSN

As mentioned in Chapter 1, suitable network architecture is important to support efficient information communication and integration in DMFSA/DMFSN. Figure 3.1 shows the proposed network architecture for DMFSA/DMFSN. Cluster architecture (CLA) is chosen because it overcomes the major limitations of both COA and HIA, by limiting the number of communication channels while at the same time tolerating possible sensor node and link failures. DMFSA/DMFSN are divided into *sensor cluster units* (SCUs), each of which consists of a set of intelligent micro flow-sensor nodes, and a base station. The intelligent sensor nodes within the same SCU are also called *sibling nodes*. Each intelligent micro flow-sensor node consists of a processing unit and an associated micro flow-sensor that measures the variables of interest. A sink node is needed for each cluster to integrate information from all the sensor nodes in the cluster. Considering the complexity of computation and/or energy constraint of the micro sensor node, a more powerful, dedicated processor is chosen as the base station for each cluster. The base station acts as the control node of the respective SCU, controlling the signal transmission of the sensor node, managing information rerouting in case of link failures, and integrating the information from all the sensor nodes in the SCU. In order to increase

the system tolerance to link failures, the sensor node sends information not only to the base station, but also to one or more other sibling nodes that are also called its *backup nodes*. Two communication protocols can be used among the sibling nodes:

(1) BP, in which each sensor node broadcasts its information to all its sibling nodes;

(2) *Evenly Wide Gossiping Protocol* (EWGP), in which instead of sending information to only one other sibling node by random selection (Heinzelman et al., 1999), the backup nodes are chosen in a way that the roles of backup nodes are evenly distributed among the sibling nodes. The intention is that no nodes are assigned too many 'duties', while others are 'starved' (Liu et al., 2001c).

The selection of protocol type and the number of backup nodes, N_b, in EWGP depends on the link failure rate, r_l, of the system. If r_l is relatively high, BP and EWGP with larger value of N_b are preferred, because they have better fault tolerance. Otherwise, EWGP with smaller value of N_b is better considered, because less communication is needed. The comparison of the communication cost of BP and EWGP with N_b equal to one is shown in section 4.2.1 in chapter four. The same protocol works for the communication among the base stations. The external processor is the commander of the whole network, in which information from all the SCUs is integrated for decision making as required. The clusters can be deployed along the flow path, depending on the flow situation and applications.

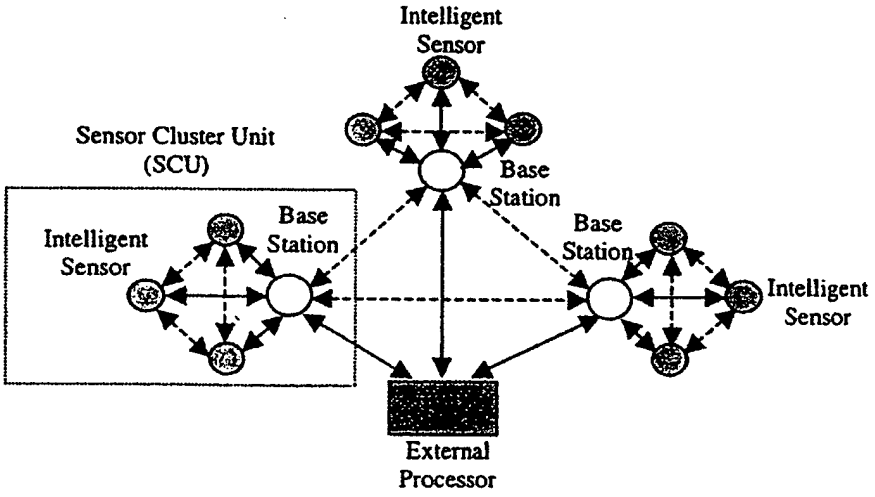


Figure 3.1 Proposed Cluster Architecture for the DMFSA/DMFSN
(Dash line means the link may or may not exist depending on the protocol used)

3.2. Fault-tolerant Time-out Communication Protocol (FTTP)

The design of this new protocol is motivated by the combination of time-out and task coordination protocols (Esfarjani and Nof, 1998), and the need for fault-tolerant measurement integration. First, the base station broadcasts the data transmission request to all the sensor nodes in the SCU. Receiving the request, each sensor node sends its data to the buffer inside the base station. Each sensor node has a corresponding buffer in the base station to store its data so that the base station can trace the source of the data. Then the data will be retrieved into the processing unit in the base station. Fault-tolerant Time-out Communication Protocol (FTTP) is an improvement over the conventional time-out protocol by incorporating it with the fault-tolerant property (Liu and Nof, 2001b). According to the time-out protocol, the base station will stop waiting for the information from the sensor node if a certain amount of time, T, has passed, where T is the pre-determined time-out period and its value depends on the application. Base station will then announce the rerouting task message to its sibling nodes. Two alternatives exist for where the base station should send the task announcement: (1) with a Blind Base Station, the message is sent to all the sibling nodes; (2) with a Smart Base Station, the message is only sent to the sibling node that can finish the task earliest, based on the current status of the nodes in the system.

3.2.1. FTTP with Blind Base Station

In this protocol, when the link of a sensor node, say node v_k, fails, the base station will broadcast the rerouting task message (RTM) for v_k to all its sibling nodes. Receiving RTM, its backup node b_{ki} (the ith backup node of v_k) generates a bid value bv_{ki}. $b_{ki} \in BK_k$, where BK_k is the set of backup nodes of v_k. bv_{ki} can be calculated as follows:

$$bv_{ki} = (m_{ki} + 1)\gamma_{ki} \qquad (1)$$

where γ_{ki} is the transmission rate of node b_{ki} (time spent in transmitting one message) and m_{ki} is the number of tasks that have been offered to node b_{ki}. Because of the computation limit of the intelligent sensor node, it is unrealistic for it to keep track of its transmission

rate over time, γ_{ki} is assumed to be constant for node b_{ki}. The base station then evaluates the priorities of the backup nodes based on their bid values (the lower the bid value, the higher the priority), and sends the task offer message (TOM) to the backup node with the highest priority, $b_{k(1)}$. Receiving the task offer message, $b_{k(1)}$ sends the required backup data message (BDM) to the base station.

The structures of the base station and intelligent nodes, and the logic chart of the FTTP with blind base station are shown in Figure 3.2 and Figure 3.3 respectively.

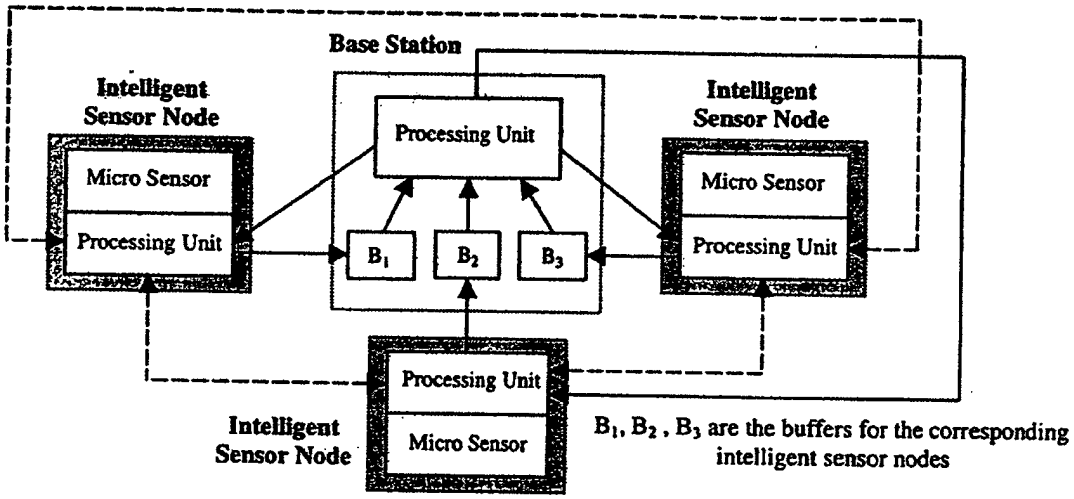


Figure 3.2 Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Blind Base Station
(Dash line means the link may or may not exist depending on the protocol used.)

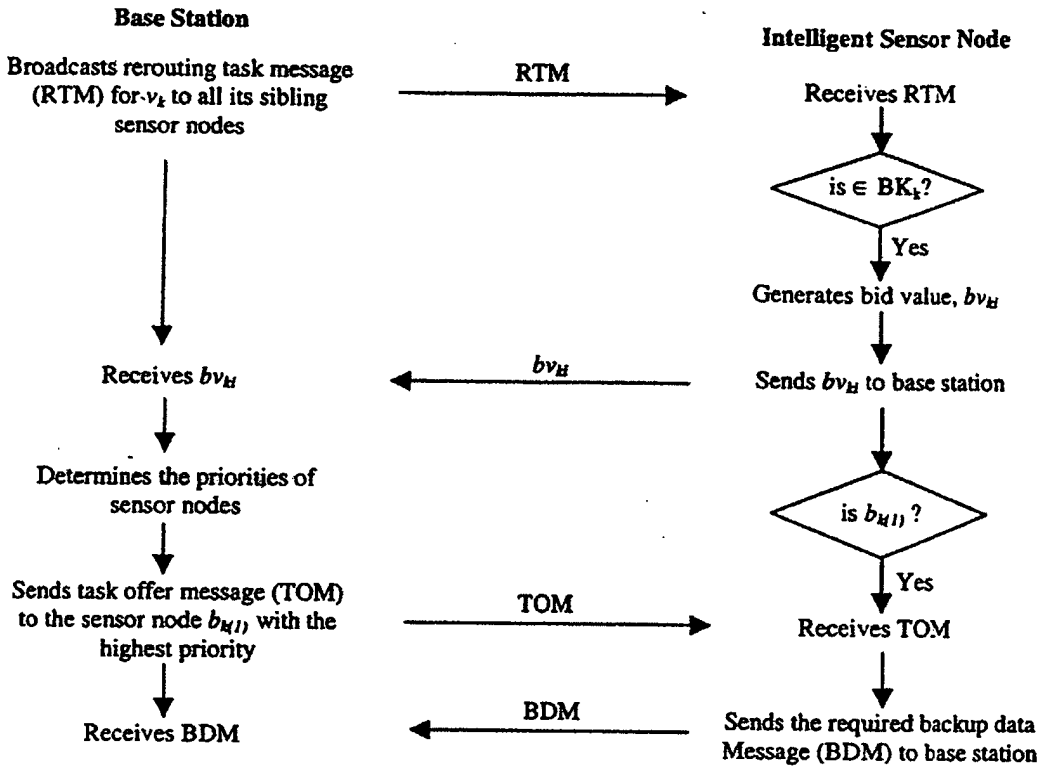


Figure 3.3 Logic Chart of the FTTP with Blind Base Station

3.2.2. FTTP with Smart Base Station

In this protocol, in order to limit the unnecessary communication and reduce the communication traffic, instead of broadcasting RTM for v_k, whose link fails, to all its sibling nodes, the base station first evaluates the priorities of the its backup nodes and then sends TOM only to the node that can finish the task the earliest. The time when the backup node b_{ki} finishes the task, t_{ki}, can be calculated as:

$$t_{ki} = (m_{ki} + 1)\gamma_{ki} \quad (2)$$

where m_{ki} is the number of tasks that have been offered to the backup node b_{ki}, and γ_{ki} is the transmission rate of b_{ki} (time spent in transmitting one message). The base station

then sends TOM to the node $b_{k(1)}$ whose $t_{k(1)}$ is the smallest. In order to find out t_{ki}, however, the base station needs to have a knowledge base which keeps track of the information for each node – its backup nodes, transmission rate, and the number of tasks that have been offered. Figure 3.4 shows the structures of base station and intelligent sensor nodes for the FTTP with smart base station. The model of the knowledge base is shown in Figure 3.5, in which n is the number of sensor nodes in the cluster and the backup nodes are hypothetical for illustration only. Figure 3.6 shows the logic chart of the FTTP with smart base station.

In comparison, relative to the FTTP with blind base station, the FTTP with smart base station requires less communication, and has the advantage of being able to keep track of the updated transmission rate of each sensor node, but it also requires a base station with more complex structure (with an additional knowledge base). Therefore, a tradeoff has to be considered in order to determine when to apply each protocol. The experimental results of the comparison of C for the FTTP with blind base station versus the FTTP with smart base station are analyzed in section 4.2.2 in chapter four.

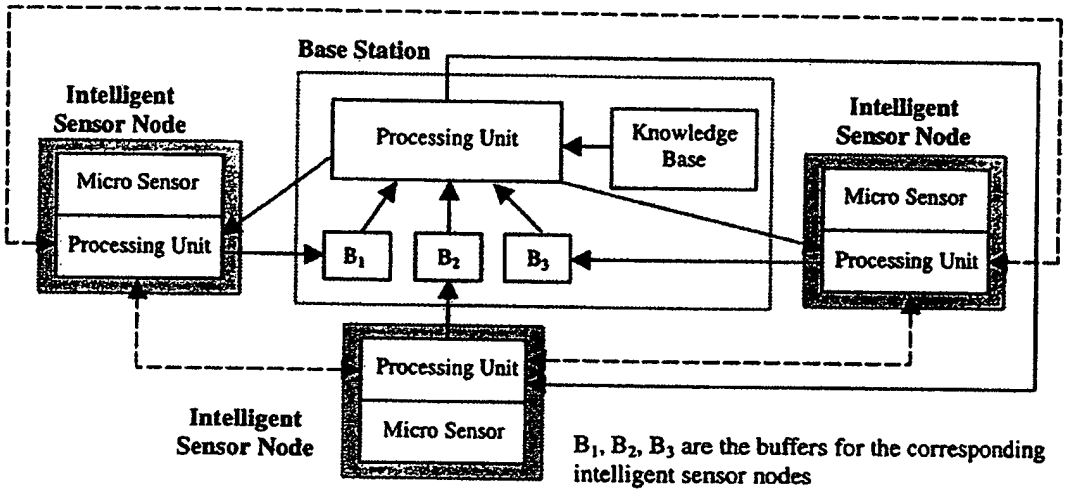


Figure 3.4 Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Smart Base Station

(Dash line means the link may or may not exist depending on the protocol used.)

V	BK	γ	m
v_1	2, 3, 4	γ_1	m_1
v_2	1, 4, 5	γ_2	m_2
...	...	...	...
v_n	3, n-2, n-1	γ_n	m_n

Knowledge Base

Figure 3.5 Model of the Knowledge Base for the FTTP with Smart Base Station

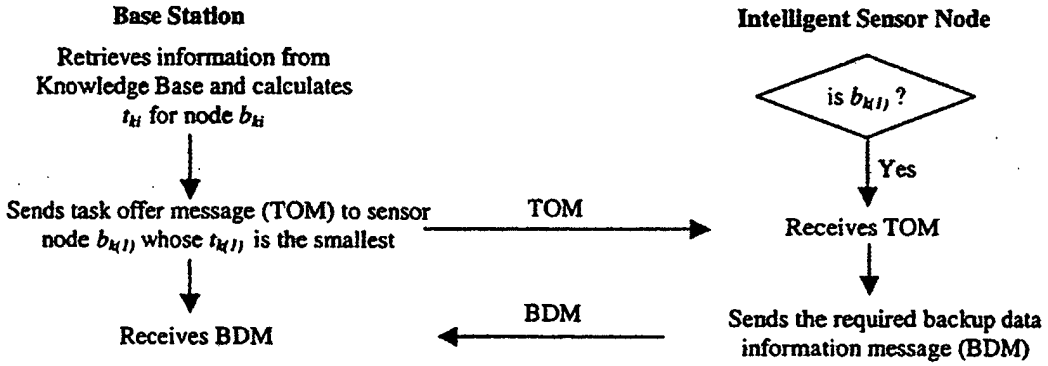


Figure 3.6 Logic Chart of the FTTP with Smart Base Station

CHAPTER 4 TIE (TEAMWORK INTEGRATION EVALUATOR)/ MEMS

TIE (Teamwork Integration Evaluator)/MEMS is developed in this thesis as a new version of the TIE simulators developed by the PRISM group (Khanna and Nof, 1994; Huang and Nof, 1996; Anussornnitisam and Nof, 2000.) The objective of TIE/MEMS is to simulate and evaluate the architectures and communication protocols developed for the distributed micro sensor network. TIE/MEMS is programmed with MPI (Message Passing Interface). MPI is not a new programming language; rather it is a library of subprograms that can be called from C and Fortran 77 programs. Since its completion in June 1994, MPI has received widespread acceptance because it is based on message passing, one of the powerful and widely used paradigms of programming parallel systems (Snir et al., 1996). MPI can be used to simulate different communication schemes among the sensor nodes, such as point-to-point communication, collective communication, group communication, and so on. In addition, implementations of MPI on top of the standard Unix inter-processor communication enable portability to multiple processors, workstation clusters, and heterogeneous networks of workstations (NOW) (Liu and Nof, 2001a).

4.1. Structure of TIE/MEMS

The overall structure of TIE/MEMS is depicted in Figure 4.1. The inputs of TIE/MEMS are:

(1) micro sensor network architecture, *NA*;

(2) communication protocol, *CP*;

(3) sensor network parameters (such as the number of micro sensor nodes, N_s, the number of sensor nodes in each cluster, N_c, the number of failed links, N_l, etc.);

(4) constraints unique to the given MEMS sensors, e.g., physical and chemical constraints causing micro-stress, proximity noise, and other disturbances to measurement and communication.

TIE/MEMS consists of a collection of libraries, each of which is built to simulate one type of sensor network architecture and communication protocol. After the required inputs are entered, the corresponding library will be called to execute the program. The output of TIE/MEMS is the communication cost, C, which is represented by the communication time c_t, or/and energy consumption c_e. If C is too high, *NA*, or *CP*, or both need to be revised. TIE/MEMS can be extended by adding libraries for the newly developed *NA* and *CP*. Table 4.1 lists some examples of constraints related to MEMS sensor operations.

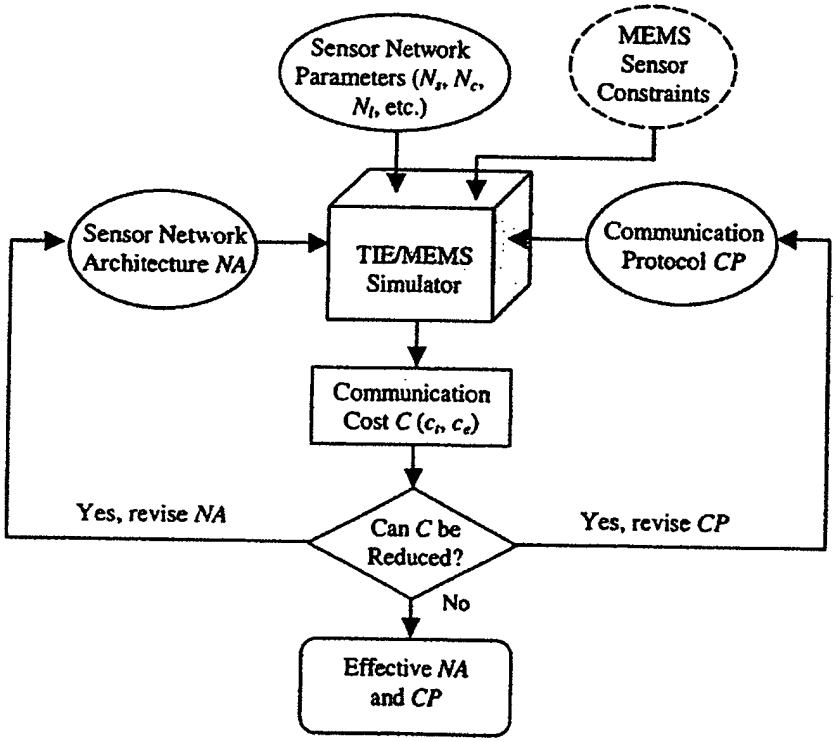


Figure 4.1 Structure of TIE/MEMS
(Dashed line – future development)

Table 4.1 Examples of Constraints Related to MEMS Sensors

Factors	Examples
Scaling issues	Viscosity force; Surface effect
Problems in micromachining	Squeeze film damping; Particular contamination; Static stiction
Response	Decrease of response time due to protective gels, coatings or diaphragms
Reliability of microelectronic device	Breakdown of a wire-bond to a junction; Defective encapsulation; Defective semiconductor material; Thermal runaway
Life time	Aging; Contamination of the device; Limited power supply
Noise/interferences	Degraded signal due to the variance of temperature, humidity, etc.

4.2. Simulation Experiments and Results

In order to illustrate how TIE/MEMS is used to evaluate the efficiency of the network architectures and communication protocols discussed in the previous chapters, two sets of experiments were conducted. The experiments were conducted on the SGI Origin 2000 workstation at the Department of Computer Science in Purdue University. The flow diagram of TIE/MEMS programming structure is shown in Figure A.1 in Appendix A. The data results of the two experiment sets are shown in more detail in Tables A.1 – A.3 in Appendix A.

4.2.1. First Experiment Set

In this experiment set, five combinations of network architectures and communication protocols were simulated and compared. They are HIA and PTP, COA and PTP, COA and BP, CLA and BP and CLA and EWGP with the number of backup nodes, N_b, equal to one. Thirty experiments were run for each combination and the outputs of the experiment are c_t and c_e per communication round. The results of c_t vs. N_s and the normalized communication time, c_{nt}, vs. N_s (normalization by the ratio to c_t of HIA and PTP) are shown in Table 4.2 and Table 4.3, respectively. Figure 4.2 shows the results of

c_{nt} vs. N_s graphically. Table 4.4 and 4.5 show the results of c_e vs. N_s (assuming the transmitting cost is 600mv per message transmission and the receiving cost is 200mw per message reception) and the normalized energy consumption, c_{ne}, vs. N_s (normalization by the ratio to c_e of the leaf node of in HIA) respectively.

Table 4.2 Communication Time (Seconds) of Five Combinations of Network Architectures and Communication Protocols vs. No. of Sensor Nodes

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	0.000102	0.000441	0.000782	0.000902	0.001674
COA	PTP	0.000725	0.006153	0.022920	0.040894	0.093200
COA	BP	0.000471	0.002233	0.004835	0.009217	0.020555
CLA	BP	0.000138[1]	0.000161[2]	0.000192[3]	0.000205[4]	0.000242[5]
CLA	EWGP (N_b=1)	0.000112[1]	0.000118[2]	0.000115[3]	0.000115[4]	0.000121[5]

[1] N_c=3; [2] N_c=4; [3] N_c=5; [4] N_c=5; [5] N_c=6

Table 4.3 Normalized Communication Time of Five Combinations of Network Architectures and Communication Protocols vs. No. of Sensor Nodes (Normalization by the ratio to Communication Time of HIA and PTP)

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	1	1	1	1	1
COA	PTP	7.143842	13.946657	29.307476	45.319050	55.668379
COA	BP	4.637767	5.061201	6.182039	10.214362	12.277506
CLA	BP	1.355993[1]	0.364110[2]	0.245972[3]	0.227731[4]	0.144386[5]
CLA	EWGP (N_b=1)	1.103448[1]	0.267473[2]	0.147046[3]	0.127443[4]	0.07234[5]

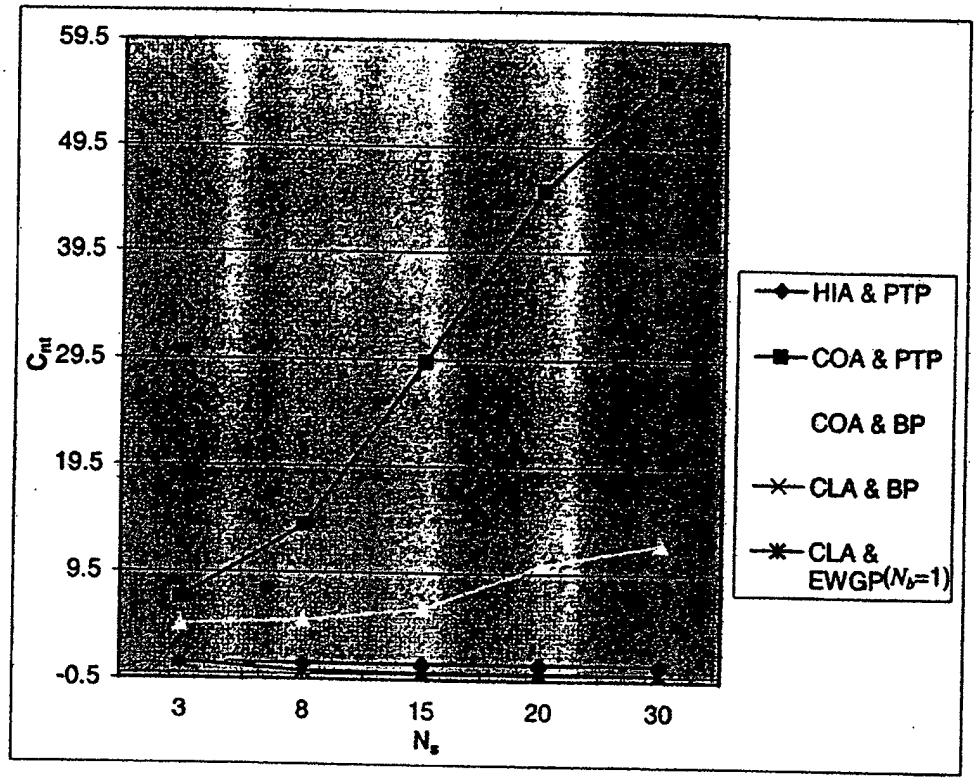


Figure 4.2 Normalized Communication Time of Five Combinations of Network Architectures and Communication Protocols vs. No. of Sensor Nodes

Table 4.4 Energy Consumption of Five Combinations of Network Architectures and Communication Protocols (mw) vs. No. of Sensor Nodes
(* c_e for the leaf nodes ; ** c_e for the intermediate nodes ; *** c_e for the root node; no intermediate nodes for HIA with when N_s is equal to 3)

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	600*	600*	600*	600*	600*
		N/A	800**	1200**	1400**	1400**
		400***	800***	800***	800***	1600***
COA	PTP	4800	44800	168000	304000	696000
COA	BP	2400	15400	50400	87400	191400
CLA	BP	1000[1]	1200[2]	1400[3]	1400[4]	1600[5]
CLA	EWGP (N_b=1)	800[1]	800[2]	800[3]	800[4]	800[5]

Table 4.5 Normalized Energy Consumption of Five Combinations of Network Architectures and Communication Protocols vs. No. of Sensor Nodes
(Normalization by the ratio to the energy consumption of the leaf node in HIA and PTP)

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	1*	1*	1*	1*	1*
		N/A	1.333333**	2.000000**	2.333333**	2.333333**
		0.666667***	1.333333***	1.333333***	1.333333***	2.666667***
COA	PTP	8.000000	74.666667	280.000000	506.666667	1160.000000
COA	BP	4.000000	25.666667	84.000000	145.666667	319.000000
CLA	BP	1.666667[1]	2.000000[2]	2.333333[3]	2.333333[4]	2.666667[5]
CLA	EWGP (N_b=1)	1.333333[1]	1.333333[2]	1.333333[3]	1.333333[4]	1.333333[5]

The experiment results indicate that when N_s is small, there is no much difference in C (both c_t and c_e) among the five combinations of *NA*s and *CP*s. In this case, COA and CLA are preferred because of their fault-tolerant characteristics. When N_s increases, however, C of the COA network, especially, the COA with PTP, increases dramatically. Therefore, HIA and CLA are favored for large sensor networks. CLA is a good choice in either case.

4.2.2. Second Experiment Set

In this experiment set, FTTP with blind base station and FTTP with smart base station, were compared. N_c and N_l are inputs of the experiment. Thirty experiments were run and the output is c_t per communication round. Table 4.6 summarizes the experiment results, which show, as expected, that the FTTP with blind base station requires less evaluating time, c_{et}, than the FTTP with smart base station. The reason for this is that in the FTTP with blind base station, the bid values are calculated by the intelligent sensor nodes locally before they are sent to the base station, and the base station only evaluates the bid values and chooses the sensor node with the minimal bid value. Much more information transmission, including sending and receiving information between the sensor nodes and the base station, however, is involved in the FTTP with blind base station. Because the information transmission time, c_{tt}, is much longer than c_{et} which requires only

computation, c_t, which is the sum of c_{et} and c_{tt}, of the FTTP with blind base station is longer than that of the FTTP with smart base station. The difference of c_t becomes more evident when N_c and N_l increase. On the other hand, the structure of the smart base station is more complex than the blind base station, as shown in Figure 3.2 and Figure 3.3. Figure 4.3 shows the ratio of c_t of the FTTP with blind base station to the FTTP with smart base station, r_{bs}, vs. N_c when N_l equal to 1, 2 and 3.

Table 4.6 Communication Time, c_t, of the FTTP with Blind Base Station and FTTP with Smart Base Station vs. No. of Sensor Nodes in the Cluster, N_c , and No. of Failed Links, N_l

FTTP	N_l	N_c	c_{et}	c_{ct}	c_t
FTTP with Blind Base Station	1	4	0.00000217	0.00014277	0.00014493
		8	0.00000270	0.00026113	0.00026383
		15	0.00000290	0.00047737	0.00048027
		21	0.00000393	0.00077307	0.00077700
	2	4	0.00000313	0.00034660	0.00034973
		8	0.00000420	0.00058190	0.00058610
		15	0.00000587	0.00114650	0.00115237
		21	0.00000740	0.00170613	0.00171353
	3	4	0.00000463	0.00055680	0.00056143
		8	0.00000565	0.00082713	0.00083277
		15	0.00000770	0.00172153	0.00172923
		21	0.00001045	0.00258752	0.00259797
FTTP with Smart Base Station	1	4	0.00000433	0.00002243	0.00002677
		8	0.00000564	0.00002891	0.00003455
		15	0.00000716	0.00003306	0.00004023
		21	0.00000817	0.00004049	0.00004866
	2	4	0.00000790	0.00004533	0.00005323
		8	0.00000989	0.00005448	0.00006437
		15	0.00001390	0.00005850	0.00007240
		21	0.00001846	0.00006068	0.00007914
	3	4	0.00001060	0.00006213	0.00007273
		8	0.00001330	0.00006696	0.00008026
		15	0.00001683	0.00007069	0.00008751
		21	0.00002386	0.00007291	0.00009677

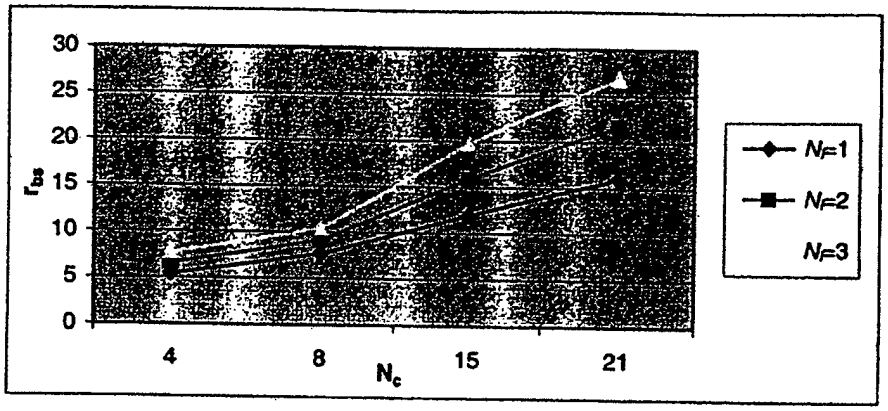


Figure 4.3 Ratio of Communication Time of the FTTP with Blind Base Station to the FTTP with Smart Base Station vs. No. of Sensor Nodes in the Cluster when the Number of Failed Links Equal to 1, 2 and 3

CHAPTER 5 FAULT-TOLERANT SENSOR INTEGRATION ALGORITHM

A system is said to be fault-tolerant if its performance is not affected by faults in the system. A new fault-tolerant sensor integration algorithm (FTSIA) is developed in this thesis to combine information from multiple, same type of sensors in the cluster and produce reliable results even if some sensors are faulty. FTSIA is applied in the base node of a sensor architecture, such as the base station in the cluster network or the root node in the hierarchical architecture. Its purpose is to combine information from the sensor nodes in the respective cluster. The algorithm developed in this research is based on the ideas originated by Marzullo (1990) and later improved by Jayasimha (1996).

5.1. Concrete and Abstract Sensors

A *concrete sensor* (CS), is a device that measures the physical variable of interest in the environment, therefore, it can be also called a physical sensor. An intelligent sensor node is a CS. An *abstract sensor* (AS), is a piecewise continuous function that maps a physical variable into a dense interval [low, high] of physical values. This interval contains the value of the physical variable read by CS. The reason for using AS is that it is more reasonable to consider the reading from a sensor as a continuous set of values instead of a point value, because of the variation of the data due to noise, manufacturing, and other uncertainty issues.

AS is derived from the point value received from CS. If a CS, say v_i, reads a value, say τ_i, and its maximum tolerable variation can be $\pm\varepsilon_i$, then AS of v_i can be defined as the

interval $[L_i, U_i]=[\tau_i - \varepsilon_i, \tau_i + \varepsilon_i]$, which is called the interval estimate of the reading from v_i, where L_i is the lower bound of v_i, and U_i is the higher bound of v_i.

5.2. Some Definitions and Terminologies

i. An AS is *correct* if its interval estimate contains the true value of the measured physical variable, otherwise, it is *faulty*. If both AS A_1 and A_2 are correct, they must intersect and the intersection contains the true value of the physical variable. An AS is said to be *tamely faulty* if it is not correct, but its interval estimate overlaps with that of a correct AS. A faulty AS that is not tamely faulty is said to be *widely faulty*.

ii. A *connected interval sequence*, S, is a sequence that has no overlaps or gaps, that is, S is made up of several intervals C_1, C_2, ..., C_k, and $u_{C_i} = l_{C_{i+1}} (1 \le i \le k-1)$, where u_{C_i} is the upper bound of interval C_i and $l_{C_{i+1}}$ is the lower bound of interval C_{i+1}.

iii. Interval I_A *completely overlaps* I_B if $l_A \le l_B$ and $u_A \ge u_B$.

iv. A distinct *m-interval* is an interval in which no more than m intervals intersect.

5.3. The Logic of FTSIA

Input: N_s ASs (L_i, U_i), $1 \le i \le N_s$, with at most f (either tamely faulty f_t, or widely faulty f_w) of them are faulty. f is a parameter determined by the manufacturer of the CSs from which the ASs were derived, based on their constraints.

Output: A final output interval and a list of all the possibly faulty sensors

Assumption: $f < \frac{1}{2} N_s$

Pseudo Program of FTSIA:

---

1. Import N_s ASs, whose lower bound set is $LB =< LB_1, LB_2, ..., LB_{N_s - f}, ..., LB_{N_s} >$ and higher bound is $HB =< HB_1, HB_2, ..., HB_{N_s - f}, ..., HB_{N_s} >$;

2. Sort the ASs by their lower bounds, and this sorted list is $L =< L_1, L_2, ..., L_{N_s-f}, ..., L_{N_s} >$ and the corresponding upper bound set is $U' =< U_1', U_2', ..., U_{N_s-f}', ..., U_{N_s}' >$;

3. Sort the first (N_s -f-1) intervals of U' and this sorted list is U ;

4. Initialize the variables

$I := \Phi$; /* Initialize the interval set, in which each interval has three attributes, lower bound l, higher bound u and level of interval m */
last := 0 ; /* Initialize the number of intervals in I */

5. Main part of the algorithm

for $j :=$ f down to 0 do

5.1 Insert U_{N_s-j}' in its appropriate place in U to keep U sorted ;

5.2 Find the intersection of ($N_s - f$) intervals !

if ($L_{N_s-j} \le U_{(1)}$), then /* $U_{(1)}$ is the first element in U */
$N := [L_{N_s-j}, U_{(1)}]$;

5.3 Insert $N := [L_{N_s-j}, U_{(1)}]$ which is a ($N_s - f$)-interval, into I and make appropriate changes to I

5.3.1Locate the connected interval sequence $S = [C_1, C_2, ..., C_k]$ in I in which each C_i intersects with N;

5.3.2 Condition 1: If S is empty, insert N to the tail I

if ($S = \Phi$)
$I := I \cup N$;
last := 1 ;

else

5.3.3 Condition 2: If there is only one interval C_1 is S, find the intersection of N and C_1

if ($S = [C_1]$)
if ($l_{c_1} = L_{N_s-j}$)
if ($U_{(1)} = u_{c_1}$) then
$m_{c_1} := m_{c_1} + 1$;
else /* if ($U_{(1)} > u_{c_1}$) */
$N_1 := [l_{c_1}, u_{c_1}]$;
$N_2 := [u_{c_1}, U_{(1)}]$;
$m_{N_1} := m_{c_1} + 1$;
$m_{N_2} := N_s - f$;
$I := (I \cup N_1 \cup N_2) - C_1$;
last := *last* + 1 ;
end if
else /* if ($l_{c_1} < L_{N_s-j}$) */

if ($U_{(1)} = u_{c_1}$) then
$N_1 := [l_{c_1}, L_{N_s-f}]$;
$N_2 := [L_{N_s-f}, U_{(1)}]$;
$m_{N_1} := N_s - f$;
$m_{N_2} := m_{c_1} + 1$;
$I := (I \cup N_1 \cup N_2) - C_1$;
$last := last + 1$;
else /* if $U_{(1)} > u_{c_1}$ */
$N_1 := [l_{c_1}, L_{N_s-f}]$;
$N_2 := [L_{N_s-f}, u_{c_1}]$;
$N_3 := [u_{c_1}, U_{(1)}]$;
$m_{N_1} := N_s - f$;
$m_{N_2} := m_{c_1} + 1$;
$m_{N_3} := N_s - f$;
$I := (I \cup N_1 \cup N_2 \cup N_3) - C_1$;
$last := last + 2$;
end if
end if
end if

else

5.3.4. Condition 3: intervals $C_2, \ldots, C_{p-1}$ completely overlaps N

for $k := 2$ to $p-1$ do
$m_{c_k} := m_{c_k} + 1$;
end for
if ($l_{c_1} = L_{N_s-f}$) then
$m_{c_1} := m_{c_1} + 1$;
else /* if $l_{c_1} < L_{N_s-f}$, Split $[l_{c_1}, u_{c_1}]$ into two intervals */
$N_1 := [l_{c_1}, L_{N_s-f}]$;
$N_2 := [L_{N_s-f}, u_{c_1}]$;
$m_{N_1} := m_{c_1}$;
$m_{N_2} := m_{c_1} + 1$;
$I := (I \cup N_1 \cup N_2) - C_1$;
$last := last + 1$;
end if;
if ($U_{(1)} < u_{c_p}$) then
$N_3 := [l_{c_p}, U_{(1)}]$;
$N_4 := [U_{(1)}, u_{c_p}]$;
$m_{N_3} := m_{c_p} + 1$;
$m_{N_4} := m_{c_p}$;

$I := (I \cup N_3 \cup N_4) - C_p$;
$last := last + 1$;
else /* if $U_{(1)} \geq u_{c_p}$ */
$m_{c_p} := m_{c_p} + 1$;
if ($U_{(1)} > u_{c_p}$) then
$N_5 := [u_{c_p}, U_{(1)}]$;
$m_{N_5} := N_5 - f$;
$I := (I \cup N_5) - C_p$;
end if
end if
end if
end if

5.3.5 Remove the AS $[L_{(1)}, U_{(1)}]$ from the list of ASs

$L := L - \{L_{(1)}\}$; $U = U - \{U_{(1)}\}$;
end for

6. Find out all the possibly faulty sensors, print out the faulty sensor set and remove it from the lower bound set and higher bound set
for $i := 0$ up to $last - 1$ do
for $j := 0$ up to N_s do
if (($HB_j >= u_{c_i}$) or ($LB_j <= l_{c_i}$)) /* sensor j is possibly faulty */
$CLB := LB - \{LB_j\}$; $CUB := UB - \{UB_j\}$; /*CLB and CUB are lower bound and higher bound sets that consist of absolutely correct sensors */
print: sensor j is possibly faulty in interval i;
end if
end for
end for

7. Find out the final output interval
sort CLB and the sorted list is $CL =< CL_1, CL_2, \ldots CL_m >$;
sort CHB and the sorted list is $CH =< CH_1, CH_2, \ldots CH_m >$;
$I_o = [CL_m, CH_1]$; /* I_o is the final data interval estimate */

The algorithm proposed by Marzullo (1990) only provides a single interval from all the sensors without fault detection. The algorithm proposed by Jayasimha (1996) provides all the (N_s-i)-interval ($0 \leq i \leq f$), but in the flow measurement applications, single output results are preferred. The improvement of the FTSIA proposed in this thesis over the two algorithms described above is that it not only detects the possibly faulty sensors and surely faulty sensors but also generates a final data interval estimate from the correct sensors after removing readings of those faulty sensors. Examples of execution of FTSIA

in different cases are shown in Appendix B. Application of FTSIA in a real flow measurement is shown in Chapter 6.

5.4. Simulation Experiments and Results

Three experiment cases were simulated to test FTSIA. The mean from FTSIA, which is the average of the higher bound and lower bound of the final interval generated from FTSIA, were compared with the statistical mean of the same measurement values. The data acquired from the FTSIA and statistical calculations are also affected by the distribution of the errors of sensor readings. In order to evaluate the reliability of both methods, the worst situation was simulated in all three cases as described below. All the sensor nodes generate readings above the true value of the measured physical variable, i.e., $\Delta\tau_i = \tau_i - \tau^* > 0$ $(i = 1,2,...,N_s)$, where τ^* is the true value of the measured physical variable, τ_i is the reading received from the sensor node v_i, and $\Delta\tau_i$ is the error of the reading from v_i. The sensor readings were simulated using the random number generator function rand( ) in Microsoft Excel. In all three cases, assuming $\tau^* = 0.5$, and the maximum tolerable variation of the sensor reading, ε, is the same for all the sensor nodes and $\varepsilon = \pm 0.005$.

Shift of values from 0.5025 to 0.505 needs to be detected because the correct sensors should give readings no greater than 0.505. Therefore, the hypotheses are:

H_0: μ=0.505

H_1: μ<0.505

Based on initial experiments, variances of both mean from FTSIA and the statistical mean of N_s sensor readings are about 3.6×10^{-5}. Assuming the type I risk is 0.025 and type II risk is 0.01, the sample size required to detect the shift is no less than one. If sample size is set to be 12 and type II risk is kept to be 0.01, then when the average of those 12 samples is less than 0.502903, reject H_0; otherwise, accept H_0 (Hicks and Turner, 1999)

5.4.1 Case 1

In this case, all the sensors are assumed to be correct, generating readings with errors within ±1% of τ^* and their interval estimates contain τ^*. Five experiment sets were conducted in this case: N_s is equal to 8, 15, 21, 27 and 36. In all the experiments of this case, H_0 is rejected for both mean from FTSIA and statistical mean of the sensor readings. Figure 5.1 shows the results from FTSIA and the statistical mean vs. N_s.

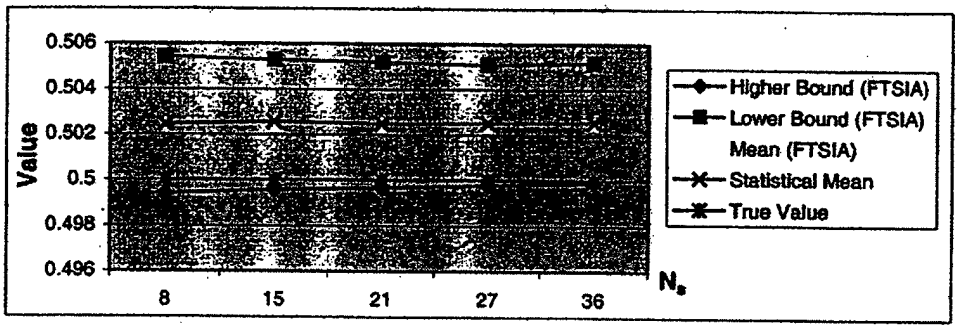


Figure 5.1 Output Interval Estimate from FTSIA and Statistical Mean vs. No. of Sensor Nodes (τ^*=0.5, ε=±0.005)

5.4.2 Case 2

In this case, some sensors are assumed to be tamely faulty, i.e., they generate readings with errors beyond ±1% but within ±2% of τ^*. Since the tamely faulty sensors intersect with some correct sensors, they can form $(N_s - f)$-intervals that do not contain the true value. Because it is impossible to know in advance which interval is the correct interval, all the possibly faulty sensors have to be deleted, and only those absolutely correct sensors will be kept. The absolutely correct sensors are those overlapping with all the $(N_s - f)$-intervals. Following this procedure, however, some correct sensors will be considered as possibly faulty sensors and thus deleted. In order for the algorithm to be effective in this case, f_t must satisfy that $f_t < \frac{1}{2}N_s$, because there will be no correct sensor left after deleting all the possibly faulty sensors when $f_t \geq \frac{1}{2}N_s$.

Twelve experiment sets were conducted in this case: N_s is equal to 8 with f_t equal to 1, 2 and 3; N_s is equal to 15 with f_t equal to 1, 3, 5 and 7; and N_s is equal to 21 with f_t equal to

1, 3, 5, 7 and 10. In this case, when N_s is equal to 8 and f_t is equal to 1 and 2, H_0 is rejected for the mean of FTSIA, but accepted for the statistical mean. When N_s is equal to 8 and f_t equal to 3, H_0 is accepted for both the mean of FTSIA and statistical mean. When N_s is equal to 15 and f_t is equal to 1, H_0 is rejected for both the mean of FTSIA and statistical mean. When N_s is equal to 15 and f_t is equal to 2 and 3, H_0 is rejected for the mean of FTSIA, but accepted for statistical mean. When N_s is equal to 15 and f_t is equal to 5 and 7, H_0 is accepted for both the mean of FTSIA and statistical mean. When N_s is equal to 21 and f_t is equal to 1 and 3, H_0 is rejected for both the mean of FTSIA and statistical mean. When N_s is equal to 21 and f_t is equal to 5, H_0 is rejected for the mean of FTSIA, but accepted for statistical mean. When N_s is equal to 21 and f_t is equal to 7 and 10, H_0 is accepted for both the mean of FTSIA and statistical mean. The output interval estimates from FTSIA and statistical mean vs. f_t when N_s is equal to 8, 15 and 21 are shown in Figures 5.2 –5.4 respectively.

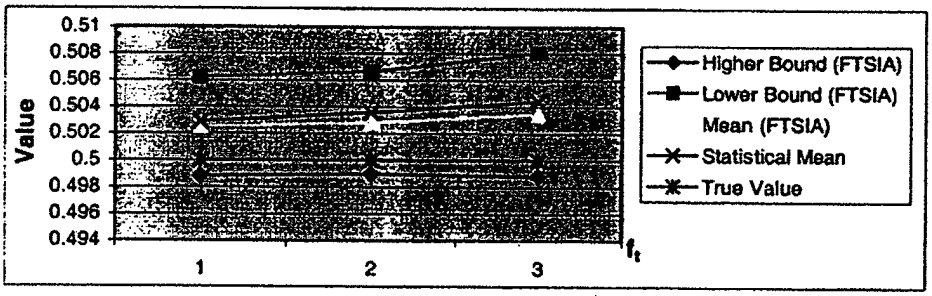


Figure 5.2 Output Interval Estimates from FTSIA and Statistical Mean vs. No. of Tamely Faulty Sensors (τ^*=0.5, ε=±0.005, N_s=8)

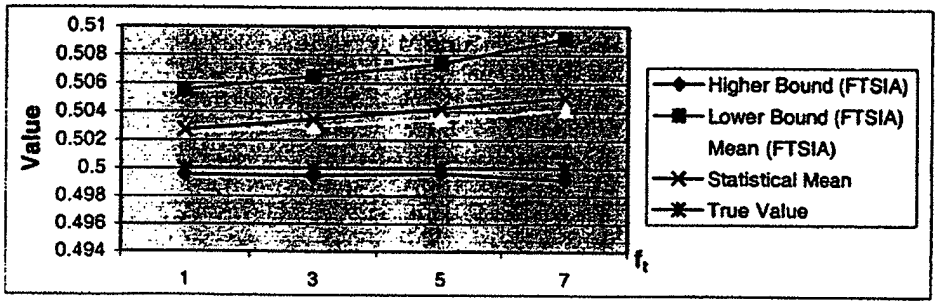


Figure 5.3 Output Interval Estimates from FTSIA and Statistical Mean vs. No. of Tamely Faulty Sensors (τ^*=0.5, ε=±0.005, N_s=15)

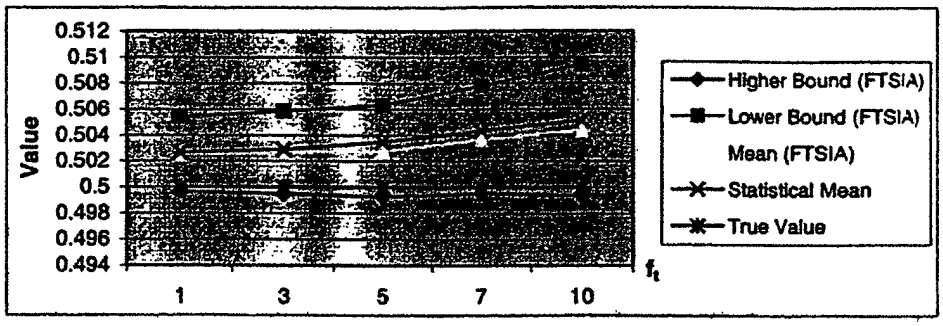


Figure 5.4 Output Interval Estimates from FTSIA and Statistical Mean vs. No. of Tamely Faulty Sensors (τ^*=0.5, ε=±0.005, N_s=21)

5.4.3. Case 3

In this case, some sensors are assumed to be widely faulty, i.e., they generate readings with errors beyond ±2% of τ^*. In order to control the experiments, the errors of the readings from the widely faulty sensors are assumed to be within ±10% of τ^*. Because when the readings from the widely faulty sensors are so close that their interval estimates intersect, they will form $(N_s - f)$-intervals that deviate the results from the correct one. In order to prevent this situation, the number of widely faulty sensors, f_w, needs to satisfy $f_w < \frac{1}{2}N_s$.

Twelve experiment sets were conducted in this case: N_s is equal to 8 with f_w equal to 1, 2 and 3; N_s is equal to 15 with f_w equal to 1, 3, 5 and 7; and N_s is equal to 21 with f_w equal to 1, 3, 5, 7 and 10.

In all the experiments in this case, H_0 is rejected for the mean of FTSIA, but accepted for statistical mean.

The output interval estimates from FTSIA and statistical mean vs. f_w when N_s is equal to 8, 15 and 21 are shown in Figures 5.5 –5.7 respectively.

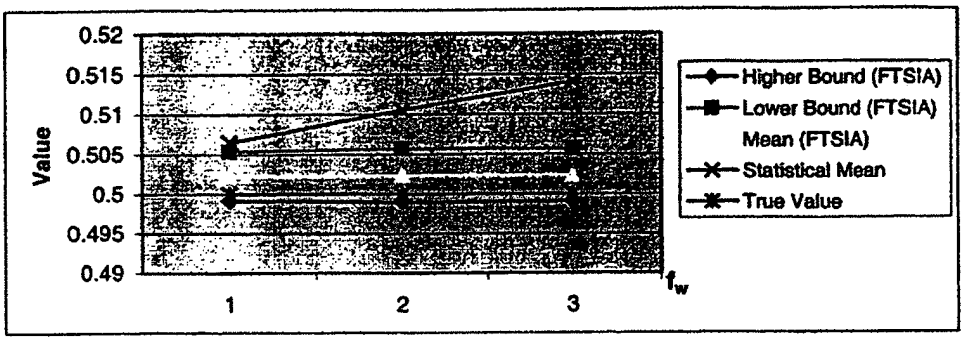


Figure 5.5 Output Interval Estimates from FTSIA and Statistical Mean vs. No. of Widely Faulty Sensors (τ^*=0.5, ε=±0.005, N_s=8)

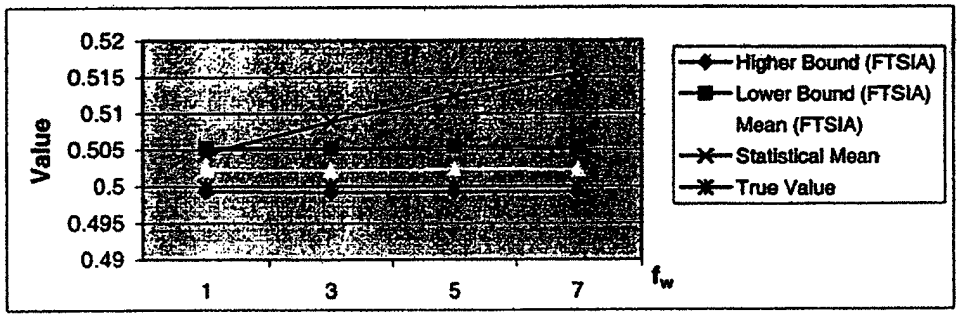


Figure 5.6 Output Interval Estimates from FTSIA and Statistical Mean vs. No. of Widely Faulty Sensors (τ^*=0.5, ε=±0.005, N_s=15)

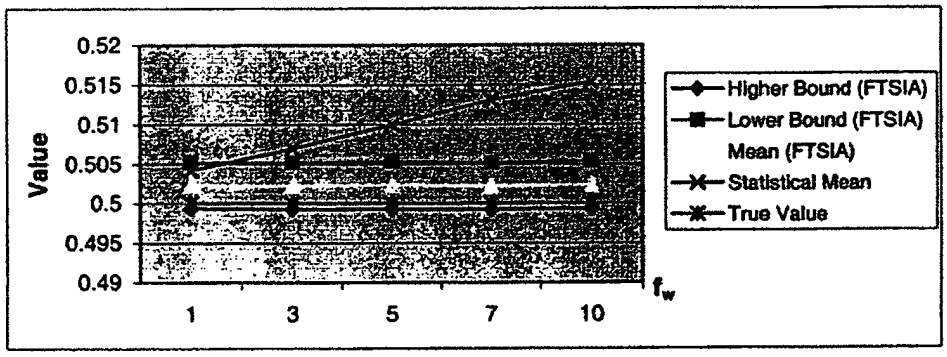


Figure 5.7 Output Interval Estimates from the FTSIA and Statistical Mean vs. No. of Widely Faulty Sensors(τ^*=0.5, ε=±0.005, N_s=21)

5.5. Conclusions of FTSIA Experiments

Analysis of Variance (ANOVA) was used to further analyze the results from the experiments in section 5.4. The mean from FTSIA and statistical mean were compared. Table 5.1 summarizes the analysis results, which show that when f=0 or f_t is small compared with N_s, there is no significant difference between the mean from FTSIA and the mean from the statistical calculations. However, when f_t increases or f_w>0, the mean from FTSIA is significantly more accurate than the statistical mean.

Table 5.2 shows how N_s and f_t affect the mean from FTSIA in case 2. The reason for this result is that tamely faulty sensors can intersect with some correct sensors and form $(N_s - f)$-intervals that do not contain the true value, and the correct sensors that do not intersect with the incorrect intervals will also be deleted as possibly faulty sensors. The correct sensors left after deletion usually have relatively higher errors because they must not only intersect with the correct sensors, but also with the tamely faulty sensors. For example, in case 2, when N_s=15 and f_t =7, and when N_s=21 and f_t =10, only one correct sensor is left after deleting all the possibly faulty sensors and this sensor has the highest sensor reading, i.e., the largest error, although still within the tolerable limits. Therefore, the performance of FTSIA will degrade gracefully when f_t increases, but the final data interval estimate it generates will still contain the true value. Table 5.2 also indicates that in order to enable FTSIA generate more accurate results, the number of correct sensors left after deletion, n_c, should be at least half of N_s, i.e., $n_c \geq \frac{1}{2} N_s$. Based on the logic of the algorithm, n_c can be calculated as

$$n_c = N_s - 2 * f_t . \tag{3}$$

Therefore, when all the faulty sensors are tamely faulty in the system, in order to generate more accurate result using the FTSIA, N_s should satisfy

$$N_s \geq 4 * f_t \tag{4}$$

For example, when f_t=1 and $N_s \geq 4$, when f_t =3 and $N_s \geq 12$, and when f_t=5 and $N_s \geq 20$, the FTSIA will yield more accurate results.

The widely faulty sensors (case 3), however, do not cause the same problem as tamely faulty sensors. They do not intersect with the correct sensors and are simpler to detect and

delete, without affecting the correct sensors (see Table 5.3). Therefore, N_s and f_w do not have significant effect on the results from FTSIA, as long as the f_w is less than half of N_s, i.e., $f_w < \frac{1}{2}N_s$. Therefore, when all the faulty sensors are widely faulty in the system, N_s should satisfy

$$N_s > 2 * f_w \tag{5}$$

Since Equations (4) and (5) are linearly independent, it can be concluded that when both tamely faulty and widely faulty sensors exist in the system, in order for FTSIA to generate more accurate results, N_s should satisfy

$$N_s > 4 * f_t + 2 * f_w \tag{6}$$

The outputs of FTSIA also infer which sensor nodes are possibly tamely faulty, and which are widely faulty. If some sensor node exists in the all the possibly faulty sensor sets, it is widely faulty. On the other hand, if some sensor node appears in some of the possibly faulty sensor sets, but not all of them, it is possibly tamely faulty.

5.6. FTSIA Algorithmic Characteristics

The simulation experiment results show that the proposed FTSIA is effective because although when f_t is large relative to N_s, and the performance of the FTSIA may degrade gracefully, it always determines the closed interval that contains the true value. This algorithm has a complexity of $O(N_s \log(N_s))$ because of the sorting in step 2 and 3 shown in the logic of the algorithm (the complexity of computing statistical mean is of $O(N_s)$). This algorithm is simple and easy to implement (section 5.3). The results of three experimental cases are shown in more detail in Tables C.1 –C.3 in Appendix C. Summary of the 29 experiments (with 12 treatments each) in all three cases is shown in Figure C.1 in Appendix C.

Given the failure rates of the sensors and the required accuracy of the results, N_s can be determined.

Table 5.1 ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Faulty Sensors on the Mean from FTSIA and Statistical Mean (From three experimental cases)

			Conclusions 1 (Mean from the FTSIA vs. Statistical Mean)	Conclusions 2 (Mean from FTSIA)
f=0			**	N_s=8 is ** from N_s=15 and ** from N_s=21**
f_f>0	f_f=1	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	**	
		N_s=21	**	
	f_f=3	N_s=8	*	N_s=15 is ** from N_s=21; N_s=8 is * from N_s=15 and N_s=21
		N_s=15	*	
		N_s=21	**	
	f_f=5	N_s=15	*	N_s=15 is * from N_s=21
		N_s=21	*	
	f_f=7	N_s=15	*	N_s=15 is * N_s=21
		N_s=21	*	
	f_f=10	N_s=21	*	
f_w>0	f_w=1	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	*	
		N_s=21	*	
	f_w=3	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	*	
		N_s=21	*	
	f_w=5	N_s=15	*	N_s=15 is ** from N_s=21**
		N_s=21	*	
	f_w=7	N_s=15	*	N_s=15 is ** from N_s=21**
		N_s=21	*	
	f_w=10	N_s=21	*	

* Significantly different
** Not significantly different

Table 5.2 ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Tamely Faulty Sensors on the Mean from FTSIA (From Case 2)

N_s	Conclusions
N_s=8	f_t=1 is ** from f_t=2; f_t=3 is * from f_t=1,2
N_s=15	f_t=1 is ** f_t=3; f_t=5 is * from f_t=1,3; f_t=7 is * from f_t=5 and from f_t=1,3;
N_s=21	f_t=1 is ** from f_t=3 and ** from f_t=5; f_t=7 is * f_t=1,3,5; f_t=10 is * from f_t=7 and * from f_t=1,3,5

* Significantly different
** Not significantly different

Table 5.3 ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Widely Faulty Sensors on the Mean from the FTSIA(From Case 3)

N_s	Conclusions
N_s=8	f_w=1 is ** from f_w=2 and ** from f_w=3
N_s=15	f_w=1 is ** from f_w=3 and ** from f_w=5 and ** from f_w=7
N_s=21	f_w=1 is ** from f_w=3 and ** from f_w=5 not ** from f_w=7 and ** from f_w=10

* Significantly different
** Not significantly different

CHAPTER 6 CASE STUDY-FLOW MEASUREMENT WITH A PRESSURE SENSOR ARRAY

This section illustrates the application of the FTSIA described in Chapter 5 through a case study – measuring the flow using a pressure sensor array.

6.1. Decision on the Number of Sensors

Before the experiment was conducted, N_s had to be decided to generate more accurate result using the FTSIA. The initial experiments showed that among the 8 available pressure sensors, f is equal to 2. Therefore, according to Equation (6), all 8 available pressure sensors had to be used to form a pressure sensor array in the experiment.

6.2. Description of the Experiment

This experiment was conducted in the PRISM lab at Purdue University. The experiment setup is shown in Figure 6.1. It consisted of a vacuum cleaner, which was used to suck air to go through the pipe connected to it, and a pressure sensor array, which had 8 pressure sensors. 8 holes were drilled on the pipe and signals were then input into the pressure sensors through the 1/8 Tygon Tubing. Because the input signals were very low (less than 0.01v), differential inputs were used to reduce the noise effect. Data Acquisition (DAQ) board was used to change the analog signals into the digital signals which were input into the computer for data analysis. In this experiment, the DAQ used was KPCI-3107, manufactured by Keithley Instruments Inc. KPCI-3107 offers some features that satisfy the experiment, such as maximum sample rate up to 100KS/s, 16 single-ended or 8 differential inputs, 12 gain ranges (1, 2, 4, 8, 10, 20, 40, 80, 100, 200,

400, 800) and compatible with LabView, etc. LabView is a graphical programming language that has been widely adopted throughout industry, academia, and government labs as the standard for data acquisition, instrument control software, and analysis software (Bishop, 2001). Finally, the data retrieved from LabView were input into the FTSIA to generate a final data interval estimate output.

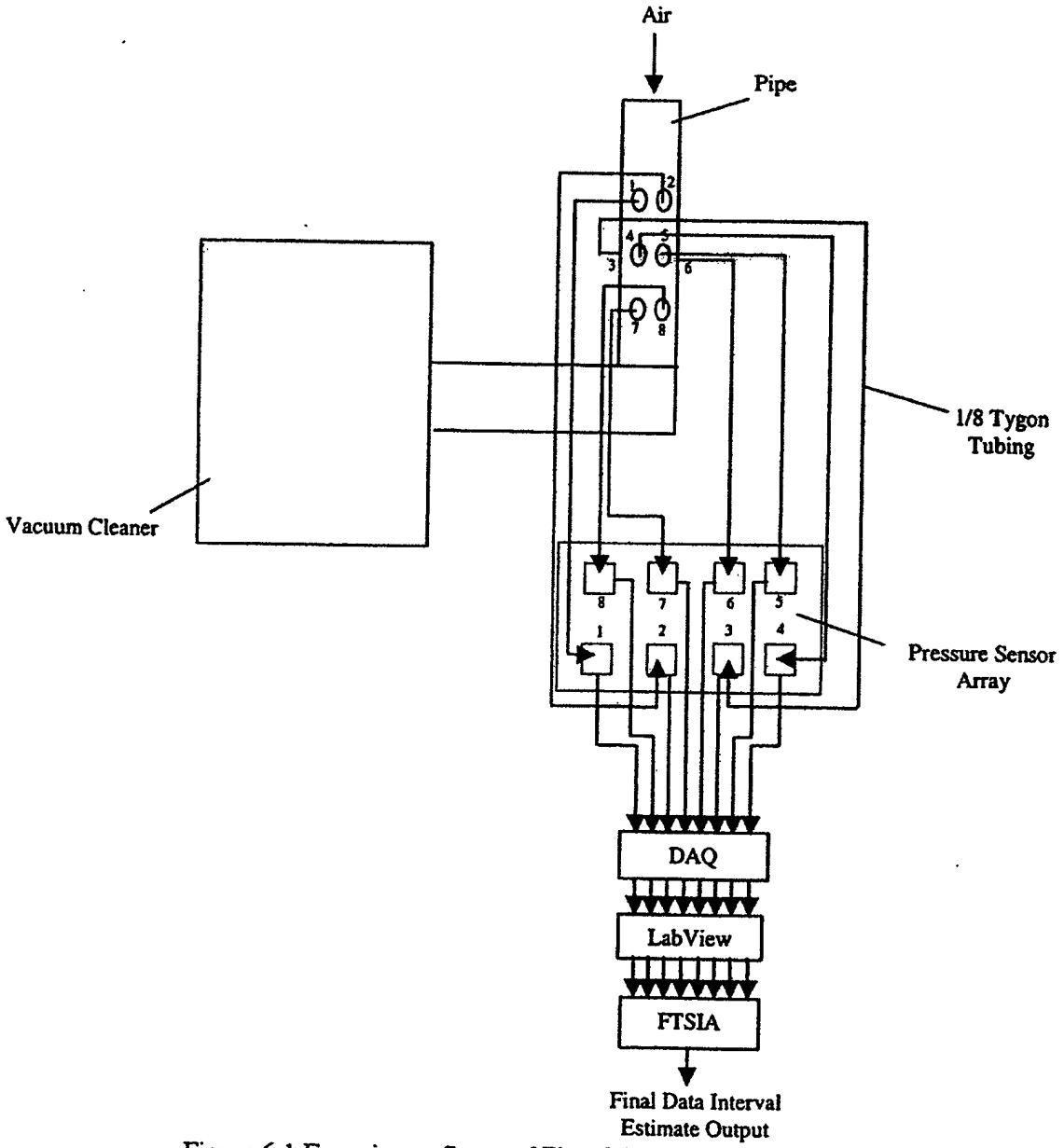


Figure 6.1 Experiment Setup of Flow Measurement with A Pressure Sensor Array

6.3. Experiment Results

Table 6.1 shows an example of the data, τ_i, received from the pressure sensor array and the lower bounds, L_i, and higher bounds, U_i, of the data intervals, assuming ε=±0.0005. After the L_i and U_i are input into the FSTIA, the outputs generated are as shown in Table 6.2. The output results show that there is an 8-interval, which means the data from all sensors intersect, two 7-intervals and one 6-interval. When there are tamely faulty sensors in the system, and if the lower bounds (higher bounds) of their data intervals are lower (higher) than the maximum (minimum) of the higher bounds (lower bounds) of the correct data intervals, it is possible that the data from all the sensors intersect, so it cannot be concluded that there is no faulty sensor in the system. Therefore, the intervals of three possibly faulty sensors, 4, 5, and 7 have to be removed from the interval set, and the final interval (the intersection of all the absolutely correct sensors) is [0.006747, 0.007685]. In addition, the results show that sensors 4, 5 and 7 are possibly tamely faulty because none of them exist in all of the possibly faulty sets and thus can be removed from the system for re-calibration or replacement.

Table 6.1 An Example of the Data Received from 8 Pressure Sensors (volt) and the Lower Bounds and Higher Bounds of the Data Interval (Assuming ε=±0.0005)

v_i	0	1	2	3	4	5	6	7
τ_i	0.007237	0.007212	0.007357	0.007158	0.007150	0.007770	0.007247	0.007132
L_i	0.006737	0.006712	0.006857	0.006658	0.00665	0.007270	0.006747	0.006632
U_i	0.007737	0.007712	0.007857	0.007658	0.00765	0.00827	0.007747	0.007632

Table 6.2 Summary of the Outputs from FTSIA (Data from Table 6.1 are inputs)

FTSIA	*m-Interval*					
	6-interval		7-interval		8-interval	
	Interval	Faulty Sensor(s)	Interval	Faulty Sensor(s)	Interval	Faulty Sensor(s)
	[0.007650,0.007685]	4,7	[0.006857,0.007270]	5	[0.007270,0.007632]	None
			[0.007632,0.007650]	7		
	Final Interval	[0.006857, 0.007685]		Mean	0.007271	
Statistic Calculation	Mean	0.0072829	Standard Deviation	0.0002095		

CHAPTER 7 CONCLUSIONS AND FUTURE RESEARCH

In this thesis, three issues of design of the distributed micro flow-sensor arrays and networks (DMFSA/DMFSN) were discussed, including the sensor network architecture, communication protocol and sensor information integration algorithm. TIE/MEMS was developed to evaluate the efficiency of the network architectures and communication protocols.

7.1. Design Recommendations of DMFSA/DMFSN

Figure 7.1 shows the overall logic of the design recommendations of DMFSA/DMFSN. In order to design DMFSA/DMFSN, first of all, the physical constraints of the micro flow-sensors used in the system have to be examined, such as the deterioration rate, r_d, link failure rate, r_l, maximum number of failed sensors, f_l, f_w, etc. Based on the information, the number of sensors required in the system, N_s, to provide more accurate results can be decided using Equation (6).

Next, the type of network architecture, *NA*, and communication protocols, *CP*, for DMFSA/DMFSN need to be chosen. If N_s is relatively small (depending on the response time and/or energy constraints in the application), COA or CLA is preferred. Otherwise, HIA or CLA is preferred. However, in both cases, if the flow environment is not steady along the path, which is always the case in the real flow measurement, CLA is better than the other alternatives because the clusters can be deployed along the path. The choices of *CP* depend on the sensor physical constraints. If r_d and r_l in system are relatively high (depending on the applications of DMFSA/DMFSN), BP and EWGP with larger number of backup nodes, N_b, are preferred. Otherwise, the EWGP with less N_b is preferred. FTTP

with smart base station is preferred to FTTP with blind base station when the communication time, c_t, is critical in the application. Otherwise, if the base station with simpler structure is required, FTTP with blind base station is preferred. These statements are based on the experimental results from section 4.2 in chapter 4.

Finally, after the data from all the sensor nodes are received, FTSIA is applied in the base node and the final data interval estimate will be generated. In addition, the FTSIA also outputs the list of all of possibly faulty sensor nodes.

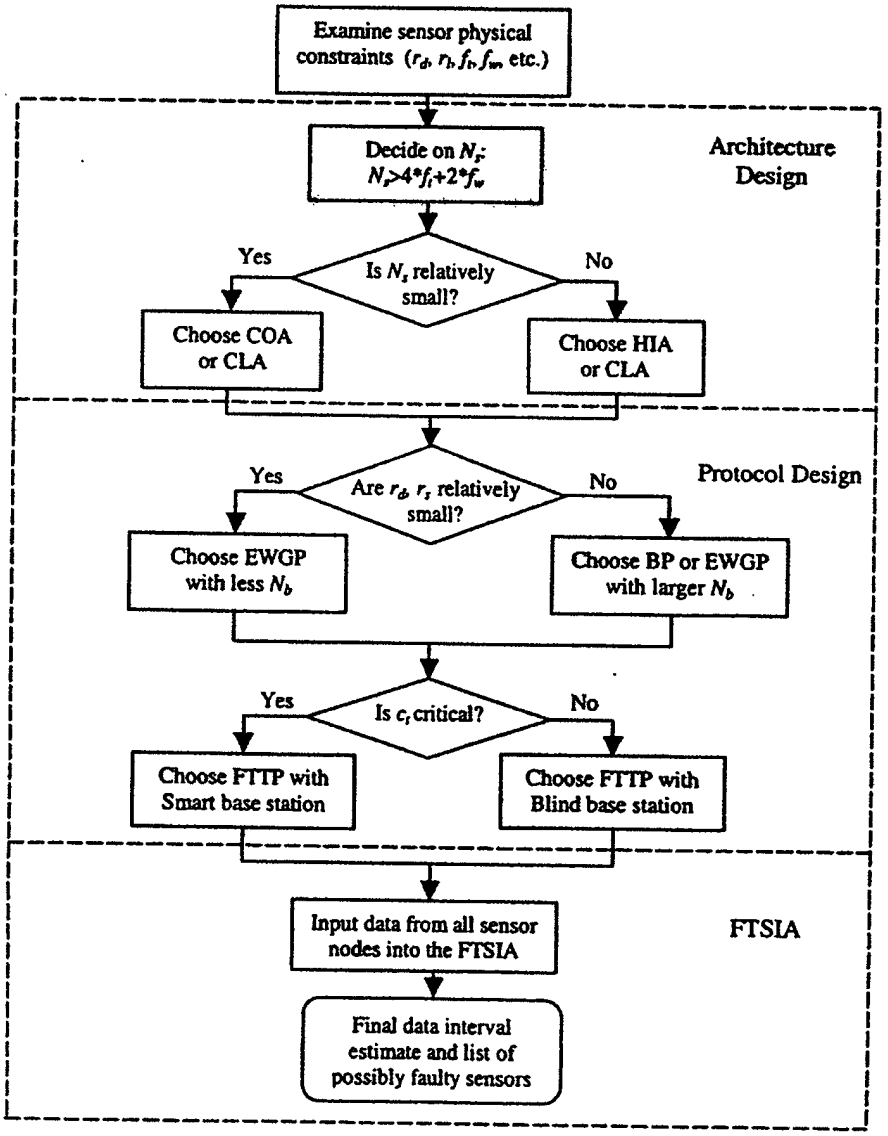


Figure 7.1 Overall Logic of Design Recommendations of DMFSA/DMFSN

7.2. Conclusions of Results

MEMS technology is the merger of the IC world with the mechanical world. It allows the integration of the sensors and actuators with IC circuitry at an unprecedented level. MEMS flow-sensor can be used in the applications in which the conventional sensor systems cannot or are not suitable to be used. DMFSA/DMFSN, built from a collection of cooperating intelligent micro-flow sensor nodes, can improve the reliability and fault-tolerance of the system.

The appropriate sensor architecture for DMFSA/DMFSN depends on the number of sensor nodes needed in the system to provide accurate information. When the number of sensor nodes in the system is relatively small, COA and CLA are better architectures because of their fault-tolerant characteristics. Otherwise, HIA and CLA are preferred in order to limit the communication cost. For example, Table 4.3 shows that when N_s is greater than 15, C_t of COA and BP is more than 10 times of C_t of HIA and PTP, and C_t of COA and PTP is more than 30 times of C_t of HIA and PTP. The proposed CLA can support efficient information communication and integration in the system, because it not only requires less communication cost, but also tolerates a certain level of sensor and link failures. For example, Table 4.3 shows that when N_s is greater than 15, C_t of CLA and BP is less than 1/4 of C_t of HIA and PTP, and C_t of CLA and EWGP (with N_b equal to 1) is less than 1/8 of C_t of HIA and PTP.

The best communication protocols depend on the number of sensor nodes in the system, sensor constraints and requirement of response time of the system. Although BP has better fault tolerance than EWGP, it requires more communication. For example, Table 4.3 shows that when N_s is equal to15 (N_c is equal to5), C_t of CLA and BP is almost twice C_t of CLA and EWGP, and the difference increases when N_s increases. Therefore, BP is suitable for the system in which the sensors and other IC components are not very robust. Compared with the FTTP with blind base station, the FTTP with smart base station requires less communication time, but it demands the base station to have an additional knowledge base that keeps track of the information of the sensor nodes. Therefore, FTTP with smart base station is preferred in the on-line control applications, in which response time is critical.

In order to evaluate the efficiency of micro sensor network architectures and communication protocols, TIE/MEMS has been developed using MPI. TIE/MEMS can be expanded by adding libraries for the newly developed network architectures and communication protocols.

Information from different sensor nodes needs to be integrated. In order to detect the faulty sensors and minimize their effects on the system, a FTSIA was proposed in this research. The results from simulation experiments show that when f_t is large relative to N_s, and when $f_w > 0$, the FTSIA gives more accurate results than statistical calculations. For example, Table 5.1 shows that when f_t is equal to 1 and N_s is equal to 8, when f_t is equal to 3 and N_s is equal to 15, and when f_w is greater than 0, the mean from FTSIA is significantly more accurate than the statistical mean. The mean from FTSIA is also affected by f_t and N_s. For example, Table 5.2 shows that when N_s is equal to 15, the mean from FTSIA when f_t is equal to 1 or 3 is significantly more accurate than that when f_t is equal to 5 and the mean from FTSIA when f_t is equal to 5 is significantly more accurate than that when f_t is equal to 7. The experiment results indicate that N_s should satisfy Equation (6) in order for FTSIA to give more accurate results.

7.3. Future Research

Three issues that have not been addressed in this thesis yet important in the design of DMFSA/DMFSN are:

1. Optimal selection and placement of sensor arrays: Optimal selection and placement of sensor arrays is not a trivial issue in the flow measurement because of several factors, such as the complex flow environment, possible interferences of the sensors, etc. Although the optimal placement depends on the flow environment, there might be some algorithms (functions) that can be applied in planning placement in the general flow environment.
2. Multi-dimensional FTSIA: A multidimensional sensor is one that returns data in D dimensions where D>1. In the flow environment, the velocity can be a 3-

dimensional vector. In such case, multi-dimensional fault-tolerant sensor integration algorithm needs to be developed.

3. Further development of TIE/MEMS: MEMS flow sensor physical constraints (as shown in Table 5.1) need to be quantified and incorporated into TIE/MEMS to evaluate the sensor network architecture and communication protocols.

LIST OF REFERENCES

LIST OF REFERENCES

1. Agre, J.R., Clare, L.P., Pottie, G.J., and Romanov, N.P., "Development platform for self-organizing wireless sensor networks", *Proceedings of SPIE - the International Society for Optical Engineering*, vol. 3713, 1999, pp.257-268.

2. Anussornnitisam, P., and Nof, S.Y., "a Teamwork Integration Evaluator for coordination protocols", *Proceedings of ICPR-2000* (CD-ROM), Bangkok, Thailand, Aug. 2000.

3. Berberig, O., Nottmeyer, K., Mizuno, J., Kanai, Y., and Kobayashi, T., "The Prandtl micro flow sensor (PMFS): a novel silicon diaphragm capacitive sensor for flow-velocity measurement", *Sensors & Actuators A-Physical*, Switzerland vol. A66, no.1-3, Apr. 1998, pp.93-98.

4. Bhushan, M., and Rengaswamy, R., 'Design of sensor location based on various fault diagnostic observability and reliability criteria", *Computers & Chemical Engineering*, UK, vol. 24, no. 2-7, 2000, pp.735-741.

5. Bishop, R.H., *Labview Student Version 6i*, Upper Saddle River, NJ, Prentice Hall PTR, 2001.

6. Brooks, R.R., and Iyengar, S.S., *Multi-sensor Fusion: Fundamentals and Applications with Software*, Upper Saddle River, NJ, Prentice Hall PTR, 1998.

7. Brown, A., and Olson, P. "Navigation and electro-optic sensor integration technology for fusion of imagery and digital mapping products", *Journal of Navigation*, UK, vol. 53, no.1, Jan. 2000, pp.132-145.

8. Chandrakasan, A., Amirtharajah, R., Seonghwan, Cho, Goodman, J., Konduri, G., Kulik, J., Rabiner, W., and Wang, A., "Design considerations for distributed microsensor systems", *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference*, Piscataway, NJ, 1999, pp.279-286.

9. Esfarjani, k., and Nof, S.Y., "A client-server model for integration and collaboration in production testing", *International Journal of Production Research*, vol. 36, no 2, 1998, pp. 3925-3321.

10. Fraden, J., *Handbook of Modern Sensors: physics, designs, and applications*, Woodbury, NY, American Institute of Physics, 1997.

11. Gass, V., van der Schoot, B.H., and de Rooij, N.F., "Nanofluid handling by micro-flow-sensor based on drag force measurements", *Proceedings of IEEE Micro Electro Mechanical Systems: An Investigation of Micro Structures, Sensors, Actuators, Machines and Systems*, Piscataway, NJ, 1993, pp.167-172.

12. Gossink, D.E., Scholz, J.B., and Gill, M.C., "Communication architecture to support distributed sensors", *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers*, Piscataway, NJ, Part 1, vol.1, 1998, pp.588-592.

13. Heinzelman, W.R., Kulik, J., and Balakrishnan, H., "Adaptive protocols for information dissemination in wireless sensor networks", *MobiCom'99, Proceedings of Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM.*, Piscataway, NJ, 1999, pp.174-185.

14. Hicks, C.R. and Turner, K.V., "Fundamental concepts in the design of experiments", Oxford University Press, NY, 1999.

15. Huang, C.Y., and Nof, S.Y., "TIE: Teamwork Integration Evaluation Simulator: A manual for TIE 1.2", *Research Memorandum*, 96-2, School of Industrial Engineering, Purdue University, 1996.

16. Ishikawa, H., Takamoto, M., Shimizu, K., Monji, H., and Matsui, G. "Sensor configuration and flow rate characteristics of ultrasonic flowmeter for very low liquids flow rate", *Transactions of the Society of Instrument and Control Engineers*, Japan, vol. 36, no.12, Dec. 2000, pp.1071-1078.

17. Iyengar, S.S., Kashyap, R.L., Madan, R.N., and Thomas, D., "Tree-structured sensor fusion architecture for distributed sensor networks", *Proceedings of SPIE-The International Society for Optical Engineering*, WA, vol. 1306, Apr.1990, pp. 126-135.

18. Iyengar, S.S., Jayasimha, D.N., and Nadig, D., "A versatile architecture for the distributed sensor integration problem", *IEEE Transactions on Computers*, vol. 43, no.2, Feb.1994, pp.175-185.

19. Iyengar, S.S, Prasad, L., and Min Hla, *Advances in Distributed Sensor Technology*, Upper Saddle River, NJ, Prentice Hall PTR, 1995.

20. Jayasimha, D.N., "Fault tolerance in multisensor networks", *IEEE Transactions on Reliability*, vol. 45, no. 2, Jun.1996, pp.308-315.

22. Khanna, N., and Nof, S.Y., "TIE, Teamwork Integration Evaluation Simulator: A preliminary User Manual for TIE 1.1", *Research Memorandum*, 94-21, School of Industrial Engineering, Purdue University, 1994.

23. Kubler, C., Raczkowsky, J., and Worn, H., "Endoscopic robots", *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2000, Third*

International Conference, Proceedings (Lecture Notes in Computer Science), Berlin, Germany, vol. 1935, 2000, pp.949-955.

24. Kuo, R.J., "Intelligent precision flexible manufacturing system through artificial neural networks and fuzzy modeling", *1998 IEEE International Joint Conference on Neural Networks Proceedings, IEEE World Congress on Computational Intelligence*, Piscataway, NJ, vol.1, 1998.

25. Leighton, F. T., *Introduction to parallel algorithms and architectures: arrays, trees, hypercubes*, San Mateo, Calif: M. Kaufmann Publishers, 1992.

26. Liu, Y., and Nof, S. Y., "TIE/MEMS: Modeling and Analysis of Micro Sensor Networks, User Manual", *Research Memorandum*, 01-01, Jan. 2001a, School of Industrial Engineering, Purdue University.

27. Liu, Y. and Nof, S.Y, "Distributed micro flow-sensor network design and modeling", *Proceedings of IFAC Workshop on Manufacturing, Modeling for Management and Control*, Prague, CR, Aug. 2-4, 2001b, pp. 161-166.

28. Liu, Y., Wereley, S. T., Nof, S.Y., and Sullivan, J.P., ""Distributed micro flow-sensor network design and modeling", PRISM Symposium, Aug. 16-18, 2001c.

29. Marzullo, K., "Tolerating failures of continuous-valued sensors", *ACM Transactions on Computer Systems*, vol. 8, no. 4, Nov. 1990, pp.284-304.

30. Mayer, F., Haberli, A., Jacobs, H., Ofner, G., Paul, O., and Baltes, H., "Single-chip CMOS anemometer", *International Electron Devices Meeting 1997, IEDM Technical Digest*, Piscataway, NJ, 1997, pp.895-898.

31. Nguyen, N.T., and Kiehnscherf, R., "Low-cost silicon sensors for mass flow measurement of liquids and gases", *Sensors & Actuators*, vol. A49, no. 1-2, Jun. 1995, pp. 17-20.

32. Paul, J.L., "Smart Sensor Web: Web-based exploitation of sensor fusion for visualization of the tactical battlefield", *IEEE Aerospace & Electronic Systems Magazine*, vol. 16, no. 5, May 2001, pp.29-36.

33. Rasmussen, A. and Zaghloul, M.E., "In the flow with MEMS", *IEEE Circuits & Devices Magazine*, vol. 14, no. 4, Jul. 1998, pp. 12-25.

34. Shofi, S., and Esashi, M., "Micro flow devices", *1994 5th International Symposium on Micro Machine and Human Science Proceedings*, Piscataway, NJ, 1994, pp.89-95.

35. Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W., and Dongarra, J., *MPI: The complete reference*, MIT press, 1996.

36. Toshio, F., and Fumihito, A., "Microrobotics", *Handbook of Industrial Robotics*, Second Ed., (Ed. S.Y. Nof), 1999, pp. 187-196.

37. Utete, S., and Durrant-Whyte, H.F., "Reliability in decentralised data fusion networks", *1994 IEEE International Conference on MFI '94, Multisensor Fusion and Integration for Intelligent Systems*, Piscataway, NJ, 1994, pp.215-221.

38. Van Kuijk, J., Lammerink, T.S.J., de Bree, H-E, Elwenspoek, M., and Fluitman, J.H.J., "Multi-parameter detection in fluid flows", *Sensors & Actuators, A-Physical*, Switzerland, vol. A47, no.1-3, Mar.-Apr. 1995, pp.369-372.

39. Von Jena, A., and Magori, V. "High-resolution ultrasonic flow meter for measuring human respiration", *Sensor 95, ACS Organisations GmbH*, Germany, 1995, pp.91-96.

40. Wesson, R., Hayes-Roth, F., Burge, W.J., Stasz, C., and Sunshine, C.A., "Network Structures for Distributed Situation Assessment", *IEEE Transaction on System*, 1981, pp. 5-23.

41. Yamasaki, H., *Handbook of Sensors and Actuators 3, Intelligent Sensors*, Yokogawa Research Institute Corporation, Tokyo, Japan, 1995.

APPENDICES

APPENDIX A TIE/MEMS Structure and Experimental Results

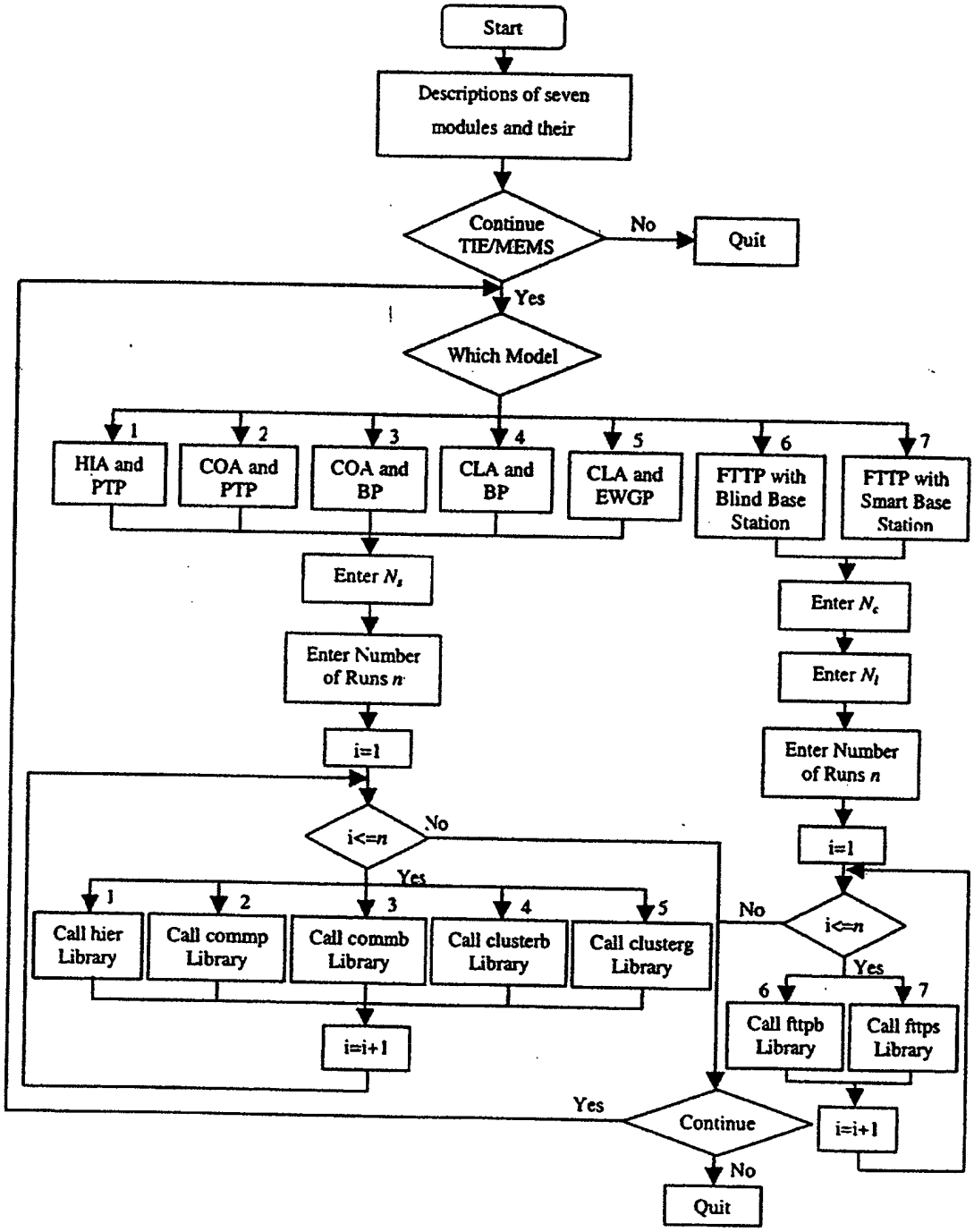


Figure A. 1 Flow Diagram of TIE/MEMS Programming Structure

Table A.1 Communication Time (Seconds) of Five Combinations of Architecture and Communication Protocol vs. No. of Sensor Nodes

a. HIA and PTP

Treatment	N_s				
	3	8	15	20	30
1	0.000129	0.000510	0.000776	0.000961	0.001843
2	0.000112	0.000496	0.000900	0.000909	0.001621
3	0.000096	0.000414	0.000726	0.000833	0.001547
4	0.000094	0.000407	0.000824	0.000891	0.001678
5	0.000087	0.000503	0.000590	0.000813	0.001803
6	0.000137	0.000469	0.000935	0.000878	0.001702
7	0.000105	0.000482	0.000844	0.001042	0.001720
8	0.000094	0.000380	0.000826	0.000872	0.001734
9	0.000064	0.000500	0.000734	0.000866	0.001658
10	0.000140	0.000407	0.000670	0.000886	0.001646
11	0.000088	0.000392	0.000826	0.001064	0.001514
12	0.000129	0.000385	0.000861	0.000806	0.001829
13	0.000150	0.000492	0.000734	0.001097	0.001734
14	0.000130	0.000509	0.000740	0.000814	0.001566
15	0.000082	0.000385	0.000645	0.000957	0.001759
16	0.000106	0.000468	0.000807	0.000858	0.001803
17	0.000074	0.000402	0.000902	0.000914	0.001751
18	0.000072	0.000402	0.000830	0.000865	0.001629
19	0.000112	0.000376	0.000737	0.000898	0.001616
20	0.000074	0.000492	0.000830	0.000868	0.001740
21	0.000100	0.000429	0.000757	0.000834	0.001730
22	0.000092	0.000482	0.000766	0.000883	0.001566
23	0.000125	0.000393	0.000880	0.000956	0.001626
24	0.000105	0.000442	0.000761	0.000885	0.001788
25	0.000070	0.000463	0.000905	0.000889	0.001716
26	0.000071	0.000468	0.000737	0.000922	0.001535
27	0.000107	0.000488	0.000857	0.000842	0.001619
28	0.000096	0.000402	0.000658	0.000938	0.001569
29	0.000111	0.000386	0.000675	0.000876	0.001476
30	0.000093	0.000411	0.000729	0.000954	0.001708

b. COA and PTP

Treatment	N_s				
	3	8	15	20	30
1	0.000878	0.005965	0.022284	0.040592	0.091560
2	0.000732	0.005984	0.022332	0.040208	0.091176
3	0.000684	0.006349	0.022356	0.041776	0.097176
4	0.000684	0.006010	0.022548	0.040032	0.095400
5	0.000655	0.006291	0.023448	0.040320	0.090840
6	0.000742	0.006221	0.023388	0.041952	0.097296
7	0.000782	0.005965	0.022572	0.040960	0.092088
8	0.000682	0.006131	0.022896	0.040848	0.090816
9	0.000679	0.006349	0.022728	0.040384	0.091464
10	0.000670	0.006298	0.023508	0.041824	0.091584
11	0.000696	0.006150	0.022416	0.040144	0.091320
12	0.000727	0.006176	0.022308	0.039952	0.097872
13	0.000797	0.005965	0.023484	0.041792	0.093360
14	0.000778	0.006106	0.023256	0.039984	0.095688
15	0.000715	0.006182	0.023388	0.040016	0.094224
16	0.000744	0.006470	0.023388	0.041536	0.092160
17	0.000665	0.005978	0.023064	0.040240	0.091224
18	0.000775	0.006061	0.022428	0.041824	0.093096
19	0.000694	0.006086	0.022668	0.041744	0.091728
20	0.000691	0.006003	0.022344	0.042160	0.094056
21	0.000694	0.006157	0.023712	0.041904	0.091656
22	0.000732	0.006630	0.022404	0.041248	0.095304
23	0.000703	0.006208	0.022692	0.042384	0.091416
24	0.000775	0.006138	0.023604	0.040128	0.092640
25	0.000698	0.006304	0.022776	0.039856	0.094632
26	0.000838	0.005933	0.022560	0.041344	0.095808
27	0.000770	0.006067	0.022728	0.040432	0.095304
28	0.000710	0.006202	0.023928	0.041280	0.091752
29	0.000684	0.006170	0.023364	0.040096	0.091176
30	0.000679	0.006035	0.023040	0.039872	0.092184

(Cont.)

c. COA and BP

Treatment	N_s				
	3	8	15	20	30
1	0.000468	0.002762	0.004802	0.009155	0.022056
2	0.000428	0.002148	0.004722	0.008859	0.019943
3	0.000437	0.002852	0.005415	0.009446	0.019346
4	0.000397	0.002091	0.004657	0.008903	0.019366
5	0.000462	0.001594	0.004786	0.008821	0.022456
6	0.000498	0.002783	0.004691	0.007229	0.019807
7	0.000482	0.002027	0.005114	0.009135	0.019613
8	0.000475	0.002172	0.004754	0.008228	0.021761
9	0.000493	0.002021	0.004045	0.009267	0.019312
10	0.000464	0.002104	0.004871	0.010808	0.023036
11	0.000450	0.002089	0.005385	0.009350	0.019392
12	0.000508	0.002485	0.004789	0.009551	0.020738
13	0.000458	0.002135	0.004774	0.009196	0.022135
14	0.000472	0.002075	0.004683	0.009016	0.019477
15	0.000475	0.002127	0.005445	0.009602	0.022655
16	0.000506	0.002069	0.005369	0.010784	0.022442
17	0.000503	0.002837	0.004607	0.008305	0.020155
18	0.000504	0.002057	0.004028	0.009043	0.021576
19	0.000510	0.002164	0.004868	0.009362	0.020248
20	0.000497	0.002128	0.004840	0.009392	0.020035
21	0.000511	0.002196	0.004576	0.009189	0.019858
22	0.000496	0.002331	0.004695	0.009237	0.018995
23	0.000500	0.002114	0.004855	0.009453	0.022536
24	0.000425	0.002821	0.004935	0.009439	0.019363
25	0.000493	0.002013	0.005092	0.009544	0.019428
26	0.000433	0.002064	0.004886	0.008925	0.018931
27	0.000461	0.002103	0.004839	0.009245	0.022214
28	0.000448	0.002221	0.004663	0.009332	0.018911
29	0.000437	0.002140	0.004995	0.009709	0.019130
30	0.000431	0.002262	0.004862	0.008988	0.021735

d. CLA and BP

Treatment	N_s				
	3	8	15	20	30
1	0.000126	0.000141	0.000183	0.000215	0.000270
2	0.000132	0.000191	0.000177	0.000235	0.000261
3	0.000145	0.000153	0.00021	0.000207	0.000225
4	0.000132	0.000150	0.000199	0.000192	0.000247
5	0.000126	0.000148	0.000208	0.000227	0.000260
6	0.000147	0.000177	0.000199	0.000206	0.000225
7	0.000136	0.000146	0.000183	0.000196	0.000265
8	0.000144	0.000188	0.000206	0.000203	0.000220
9	0.000133	0.000147	0.000174	0.000180	0.000219
10	0.000144	0.000151	0.000189	0.000210	0.000228
11	0.000163	0.000148	0.000180	0.000179	0.000235
12	0.000129	0.000156	0.000182	0.000181	0.000210
13	0.000123	0.000142	0.000196	0.000214	0.000239
14	0.000142	0.000151	0.000201	0.000215	0.000307
15	0.000134	0.000226	0.000174	0.000208	0.000242
16	0.000145	0.000157	0.000219	0.000213	0.000239
17	0.000147	0.000179	0.000172	0.000239	0.000247
18	0.000127	0.000143	0.000196	0.000177	0.000240
19	0.000142	0.000154	0.000173	0.000191	0.000231
20	0.000144	0.000154	0.000215	0.000233	0.000280
21	0.000135	0.000179	0.000180	0.000253	0.000219
22	0.000128	0.000149	0.000200	0.000180	0.000237
23	0.000129	0.000183	0.000179	0.000188	0.000247
24	0.000151	0.000153	0.000168	0.000187	0.000218
25	0.000143	0.000194	0.000202	0.000213	0.000241
26	0.000145	0.000153	0.000215	0.000195	0.000219
27	0.000142	0.000147	0.000181	0.000224	0.000242
28	0.000124	0.000141	0.000183	0.000189	0.000227
29	0.000129	0.000176	0.000236	0.000214	0.000259
30	0.000150	0.000142	0.000198	0.000204	0.000255

(Cont.)

e. CLA and EWGP ($N_b = 1$)

Treatment	N_s				
	3	8	15	20	30
1	0.000107	0.000120	0.000112	0.000128	0.000127
2	0.000106	0.000114	0.000107	0.000114	0.000118
3	0.000110	0.000136	0.000126	0.000122	0.000124
4	0.000116	0.000114	0.000117	0.000114	0.000120
5	0.000133	0.000110	0.000111	0.000116	0.000118
6	0.000113	0.000110	0.000109	0.000118	0.000119
7	0.000112	0.000114	0.000107	0.000112	0.000127
8	0.000108	0.000219	0.000106	0.000107	0.000125
9	0.000108	0.000113	0.000113	0.000118	0.000112
10	0.000105	0.000118	0.000109	0.000118	0.000120
11	0.000108	0.000118	0.000115	0.000108	0.000115
12	0.000115	0.000116	0.000118	0.000117	0.000122
13	0.000116	0.000112	0.000110	0.000124	0.000122
14	0.000108	0.000108	0.000112	0.000110	0.000115
15	0.000118	0.000112	0.000129	0.000109	0.000119
16	0.000109	0.000111	0.000111	0.000116	0.000124
17	0.000117	0.000109	0.000111	0.000112	0.000118
18	0.000109	0.000118	0.000118	0.000126	0.000119
19	0.000114	0.000144	0.000112	0.000112	0.000132
20	0.000105	0.000114	0.000117	0.000110	0.000119
21	0.000110	0.000115	0.000117	0.000115	0.000114
22	0.000108	0.000114	0.000107	0.000113	0.000121
23	0.000117	0.000111	0.000115	0.000131	0.000123
24	0.000111	0.000110	0.000122	0.000110	0.000120
25	0.000114	0.000112	0.000110	0.000112	0.000120
26	0.000108	0.000118	0.000112	0.000110	0.000135
27	0.000112	0.000105	0.000126	0.000110	0.000115
28	0.000111	0.000116	0.000113	0.000107	0.000128
29	0.000106	0.000111	0.000114	0.000120	0.000113
30	0.000109	0.000113	0.000130	0.000111	0.000121

Table A.2 Communication Time (Seconds) of the FTTP with Blind Base Station vs. No. of Sensor Nodes in the Cluster and No. of Failed Links

a. $N_c = 4, N_l = 1$

Treatment	c_{et}	c_{tt}	c_t
1	0.000003	0.000169	0.000172
2	0.000002	0.000129	0.000131
3	0.000002	0.00014	0.000142
4	0.000003	0.000157	0.00016
5	0.000002	0.000132	0.000134
6	0.000002	0.000137	0.000139
7	0.000001	0.000119	0.00012
8	0.000002	0.000128	0.00013
9	0.000002	0.000165	0.000167
10	0.000003	0.000165	0.000168
11	0.000003	0.000134	0.000137
12	0.000001	0.000163	0.000164
13	0.000003	0.000125	0.000128
14	0.000002	0.00013	0.000132
15	0.000002	0.000154	0.000156
16	0.000002	0.000146	0.000148
17	0.000002	0.000123	0.000125
18	0.000002	0.000131	0.000133
19	0.000002	0.000152	0.000154
20	0.000002	0.000162	0.000164
21	0.000002	0.000164	0.000166
22	0.000003	0.000164	0.000167
23	0.000002	0.000136	0.000138
24	0.000002	0.000126	0.000128
25	0.000002	0.00012	0.000122
26	0.000002	0.000175	0.000177
27	0.000002	0.000123	0.000125
28	0.000002	0.000128	0.00013
29	0.000002	0.000124	0.000126
30	0.000003	0.000162	0.000165

b. $N_c = 8, N_l = 1$

Treatment	c_{et}	c_{tt}	c_t
1	0.000003	0.000279	0.000282
2	0.000004	0.000269	0.000273
3	0.000003	0.000198	0.000201
4	0.000003	0.000218	0.000221
5	0.000001	0.000263	0.000264
6	0.000002	0.000252	0.000254
7	0.000002	0.000242	0.000244
8	0.000006	0.000241	0.000247
9	0.000005	0.00029	0.000295
10	0.000002	0.000305	0.000307
11	0.000003	0.000215	0.000218
12	0.000002	0.000263	0.000265
13	0.000004	0.000234	0.000238
14	0.000002	0.000227	0.000229
15	0.000002	0.000246	0.000248
16	0.000002	0.000258	0.00026
17	0.000002	0.00026	0.000262
18	0.000002	0.000211	0.000213
19	0.000004	0.000329	0.000333
20	0.000003	0.000307	0.00031
21	0.000003	0.000287	0.00029
22	0.000002	0.00027	0.000272
23	0.000003	0.000299	0.000302
24	0.000002	0.000245	0.000247
25	0.000004	0.000304	0.000308
26	0.000003	0.000241	0.000244
27	0.000002	0.000244	0.000246
28	0.000002	0.000278	0.00028
29	0.000002	0.00029	0.000292
30	0.000001	0.000269	0.00027

(Cont.)

c. $N_c = 15, N_l = 1$

Treatment	c_{et}	c_u	c_t
1	0.000002	0.000418	0.000420
2	0.000002	0.000479	0.000481
3	0.000002	0.000440	0.000442
4	0.000003	0.000495	0.000498
5	0.000002	0.000451	0.000453
6	0.000003	0.000492	0.000495
7	0.000005	0.000484	0.000489
8	0.000005	0.000467	0.000472
9	0.000003	0.000461	0.000464
10	0.000004	0.000500	0.000504
11	0.000003	0.000439	0.000442
12	0.000002	0.000452	0.000454
13	0.000003	0.000509	0.000512
14	0.000003	0.000457	0.000460
15	0.000002	0.000479	0.000481
16	0.000002	0.000540	0.000542
17	0.000002	0.000472	0.000474
18	0.000002	0.000730	0.000732
19	0.000005	0.000497	0.000502
20	0.000002	0.000532	0.000534
21	0.000004	0.000430	0.000434
22	0.000002	0.000456	0.000458
23	0.000002	0.000433	0.000435
24	0.000003	0.000494	0.000497
25	0.000004	0.000464	0.000468
26	0.000002	0.000424	0.000426
27	0.000003	0.000441	0.000444
28	0.000002	0.000476	0.000478
29	0.000005	0.000472	0.000477
30	0.000003	0.000437	0.000440

d. $N_c = 21, N_l = 1$

Treatment	c_{et}	c_u	c_t
1	0.000004	0.000663	0.000667
2	0.000003	0.000854	0.000857
3	0.000004	0.000638	0.000642
4	0.000004	0.000784	0.000788
5	0.000004	0.000826	0.000830
6	0.000006	0.000794	0.000800
7	0.000004	0.000946	0.000950
8	0.000003	0.000769	0.000772
9	0.000006	0.000751	0.000757
10	0.000003	0.000688	0.000691
11	0.000004	0.000734	0.000738
12	0.000004	0.000839	0.000843
13	0.000003	0.000691	0.000694
14	0.000003	0.000908	0.000911
15	0.000002	0.000906	0.000908
16	0.000004	0.000995	0.000999
17	0.000004	0.000734	0.000738
18	0.000004	0.000674	0.000678
19	0.000004	0.000764	0.000768
20	0.000004	0.000984	0.000988
21	0.000004	0.000595	0.000599
22	0.000004	0.000866	0.000870
23	0.000004	0.000762	0.000766
24	0.000003	0.000683	0.000686
25	0.000004	0.000744	0.000748
26	0.000004	0.000747	0.000751
27	0.000005	0.000716	0.000721
28	0.000005	0.000840	0.000845
29	0.000003	0.000575	0.000578
30	0.000005	0.000722	0.000727

(Cont.)

e. $N_c = 4, N_l = 2$

Treatment	c_{et}	c_{tt}	c_t
1	0.000003	0.000362	0.000365
2	0.000005	0.000339	0.000344
3	0.000004	0.000320	0.000324
4	0.000004	0.000358	0.000362
5	0.000004	0.000342	0.000346
6	0.000002	0.000390	0.000392
7	0.000003	0.000369	0.000372
8	0.000003	0.000376	0.000379
9	0.000004	0.000370	0.000374
10	0.000002	0.000344	0.000346
11	0.000003	0.000346	0.000349
12	0.000004	0.000346	0.000350
13	0.000002	0.000313	0.000315
14	0.000002	0.000308	0.000310
15	0.000003	0.000319	0.000322
16	0.000004	0.000384	0.000388
17	0.000004	0.000338	0.000342
18	0.000005	0.000291	0.000296
19	0.000003	0.000382	0.000385
20	0.000002	0.000353	0.000355
21	0.000005	0.000301	0.000306
22	0.000002	0.000374	0.000376
23	0.000004	0.000330	0.000334
24	0.000002	0.000339	0.000341
25	0.000003	0.000373	0.000376
26	0.000002	0.000310	0.000312
27	0.000003	0.000298	0.000301
28	0.000003	0.000371	0.000374
29	0.000002	0.000374	0.000376
30	0.000002	0.000378	0.000380

f. $N_c = 8, N_l = 2$

Treatment	c_{et}	c_{tt}	c_t
1	0.000006	0.000577	0.000583
2	0.000004	0.000652	0.000656
3	0.000004	0.000581	0.000585
4	0.000005	0.000651	0.000656
5	0.000006	0.000592	0.000598
6	0.000005	0.000513	0.000518
7	0.000005	0.000595	0.000600
8	0.000005	0.000583	0.000588
9	0.000003	0.000557	0.000560
10	0.000004	0.000586	0.000590
11	0.000004	0.000584	0.000588
12	0.000003	0.000583	0.000586
13	0.000005	0.000504	0.000509
14	0.000004	0.000541	0.000545
15	0.000005	0.000561	0.000566
16	0.000004	0.000512	0.000516
17	0.000006	0.000588	0.000594
18	0.000003	0.000623	0.000626
19	0.000003	0.000595	0.000598
20	0.000004	0.000602	0.000606
21	0.000003	0.000623	0.000626
22	0.000006	0.000550	0.000556
23	0.000003	0.000656	0.000659
24	0.000004	0.000593	0.000597
25	0.000004	0.000571	0.000575
26	0.000004	0.000541	0.000545
27	0.000003	0.000520	0.000523
28	0.000004	0.000630	0.000634
29	0.000003	0.000568	0.000571
30	0.000004	0.000625	0.000629

(Cont.)

g. $N_c = 15, N_l = 2$

Treatment	c_{et}	c_{it}	c_t
1	0.000005	0.001058	0.001063
2	0.000006	0.001044	0.001050
3	0.000003	0.001193	0.001196
4	0.000006	0.001255	0.001261
5	0.000008	0.001174	0.001182
6	0.000006	0.001050	0.001056
7	0.000006	0.001164	0.001170
8	0.000006	0.001226	0.001232
9	0.000004	0.001230	0.001234
10	0.000007	0.001298	0.001305
11	0.000004	0.001044	0.001048
12	0.000006	0.001302	0.001308
13	0.000005	0.000966	0.000971
14	0.000010	0.001284	0.001294
15	0.000007	0.001142	0.001149
16	0.000006	0.001214	0.001220
17	0.000009	0.001090	0.001099
18	0.000004	0.001038	0.001042
19	0.000005	0.001236	0.001241
20	0.000006	0.001113	0.001119
21	0.000006	0.001154	0.001160
22	0.000006	0.001182	0.001188
23	0.000005	0.000948	0.000953
24	0.000006	0.001212	0.001218
25	0.000004	0.001021	0.001025
26	0.000006	0.001194	0.001200
27	0.000006	0.001282	0.001288
28	0.000009	0.001354	0.001363
29	0.000006	0.000778	0.000784
30	0.000003	0.001149	0.001152

h. $N_c = 21, N_l = 2$

Treatment	c_{et}	c_{it}	c_t
1	0.000008	0.001726	0.001734
2	0.000007	0.001637	0.001644
3	0.000006	0.001834	0.001840
4	0.000008	0.001882	0.001890
5	0.000006	0.001507	0.001513
6	0.000006	0.001637	0.001643
7	0.000007	0.001719	0.001726
8	0.000006	0.001458	0.001464
9	0.000006	0.001649	0.001655
10	0.000009	0.001944	0.001953
11	0.000006	0.001906	0.001912
12	0.000006	0.001725	0.001731
13	0.000007	0.001499	0.001506
14	0.000008	0.001798	0.001806
15	0.000006	0.001685	0.001691
16	0.000008	0.001891	0.001899
17	0.000006	0.001625	0.001631
18	0.000006	0.001635	0.001641
19	0.000005	0.001706	0.001711
20	0.000010	0.001793	0.001803
21	0.000008	0.001799	0.001807
22	0.000009	0.001536	0.001545
23	0.000007	0.001734	0.001741
24	0.000006	0.001703	0.001709
25	0.000010	0.001579	0.001589
26	0.000010	0.001566	0.001576
27	0.000009	0.001880	0.001889
28	0.000009	0.001894	0.001903
29	0.000009	0.001615	0.001624
30	0.000008	0.001622	0.001630

(Cont.)

i. $N_c = 4, N_l = 3$

Treatment	c_{et}	c_{tt}	c_t
1	0.000005	0.000535	0.000540
2	0.000005	0.000555	0.000560
3	0.000004	0.000568	0.000572
4	0.000005	0.000571	0.000576
5	0.000004	0.000517	0.000521
6	0.000005	0.000597	0.000602
7	0.000004	0.000604	0.000608
8	0.000006	0.000627	0.000633
9	0.000005	0.000585	0.000590
10	0.000004	0.000522	0.000526
11	0.000004	0.000485	0.000489
12	0.000005	0.000554	0.000559
13	0.000005	0.000509	0.000514
14	0.000003	0.000548	0.000551
15	0.000003	0.000619	0.000622
16	0.000004	0.000553	0.000557
17	0.000005	0.000527	0.000532
18	0.000005	0.000586	0.000591
19	0.000005	0.000549	0.000554
20	0.000003	0.000539	0.000542
21	0.000005	0.000476	0.000481
22	0.000004	0.000602	0.000606
23	0.000004	0.000578	0.000582
24	0.000005	0.000549	0.000554
25	0.000006	0.000567	0.000573
26	0.000006	0.000574	0.000580
27	0.000004	0.000551	0.000555
28	0.000005	0.000575	0.000580
29	0.000006	0.000529	0.000535
30	0.000005	0.000553	0.000558

j. $N_c = 8, N_l = 3$

Treatment	c_{et}	c_{tt}	c_t
1	0.000006	0.000820	0.000826
2	0.000006	0.000836	0.000842
3	0.000006	0.000722	0.000728
4	0.000006	0.000822	0.000828
5	0.000006	0.000846	0.000852
6	0.000006	0.000837	0.000843
7	0.000004	0.000884	0.000888
8	0.000005	0.000867	0.000872
9	0.000006	0.000884	0.000890
10	0.000006	0.000824	0.000830
11	0.000007	0.000811	0.000818
12	0.000006	0.000847	0.000853
13	0.000006	0.000892	0.000898
14	0.000005	0.000814	0.000819
15	0.000005	0.000899	0.000904
16	0.000006	0.000812	0.000818
17	0.000006	0.000793	0.000799
18	0.000005	0.000875	0.000880
19	0.000005	0.000786	0.000791
20	0.000006	0.000860	0.000866
21	0.000005	0.000813	0.000818
22	0.000005	0.000721	0.000726
23	0.000006	0.000830	0.000836
24	0.000006	0.000860	0.000866
25	0.000005	0.000885	0.000890
26	0.000006	0.000858	0.000864
27	0.000006	0.000822	0.000828
28	0.000006	0.000788	0.000794
29	0.000006	0.000776	0.000782
30	0.000004	0.000739	0.000743

(Cont.)

k. $N_c = 15, N_t = 3$

Treatment	c_{et}	c_n	c_t
1	0.000006	0.001745	0.001751
2	0.000008	0.001696	0.001704
3	0.000008	0.001752	0.001760
4	0.000007	0.001640	0.001647
5	0.000008	0.001873	0.001881
6	0.000008	0.001773	0.001781
7	0.000006	0.001733	0.001739
8	0.000009	0.001811	0.001820
9	0.000008	0.001787	0.001795
10	0.000008	0.001751	0.001759
11	0.000007	0.001703	0.001710
12	0.000008	0.001718	0.001726
13	0.000007	0.001629	0.001636
14	0.000009	0.001596	0.001605
15	0.000006	0.001589	0.001595
16	0.000007	0.001839	0.001846
17	0.000009	0.001543	0.001552
18	0.000005	0.001574	0.001579
19	0.000010	0.001559	0.001569
20	0.000008	0.001737	0.001745
21	0.000010	0.001685	0.001695
22	0.000010	0.001602	0.001612
23	0.000007	0.001571	0.001578
24	0.000007	0.001922	0.001929
25	0.000006	0.001832	0.001838
26	0.000006	0.001782	0.001788
27	0.000007	0.001789	0.001796
28	0.000010	0.001740	0.001750
29	0.000006	0.001862	0.001868
30	0.000010	0.001813	0.001823

l. $N_c = 21, N_t = 3$

Treatment	c_{et}	c_n	c_t
1	0.000008	0.002769	0.002777
2	0.000010	0.002433	0.002443
3	0.000011	0.002743	0.002754
4	0.000009	0.002646	0.002655
5	0.000011	0.002777	0.002788
6	0.000010	0.002700	0.002710
7	0.000009	0.002552	0.002561
8	0.000009	0.002452	0.002461
9	0.000011	0.002550	0.002561
10	0.000010	0.002813	0.002823
11	0.000013	0.002616	0.002629
12	0.000010	0.002627	0.002637
13	0.000006	0.002510	0.002516
14	0.000009	0.002825	0.002834
15	0.000013	0.002550	0.002563
16	0.000011	0.002544	0.002555
17	0.000009	0.002608	0.002617
18	0.000010	0.002872	0.002882
19	0.000010	0.002635	0.002645
20	0.000010	0.002742	0.002752
21	0.000014	0.002538	0.002552
22	0.000014	0.002484	0.002498
23	0.000011	0.002486	0.002497
24	0.000013	0.002365	0.002378
25	0.000009	0.002533	0.002542
26	0.000011	0.002365	0.002376
27	0.000009	0.002433	0.002442
28	0.000010	0.002283	0.002293
29	0.000013	0.002587	0.002600
30	0.000024	0.002570	0.002594

Table A.3 Communication Time (Seconds) of the FTTP with Smart Base Station vs. No. of Sensor Nodes in the Cluster and No. of Failed Links

a. $N_c = 4, N_l = 1$

Treatment	c_{et}	c_{tt}	c_t
1	0.000003	0.000021	0.000024
2	0.000002	0.000016	0.000018
3	0.000006	0.000021	0.000027
4	0.000003	0.000019	0.000022
5	0.000005	0.000025	0.000030
6	0.000005	0.000021	0.000026
7	0.000005	0.000017	0.000022
8	0.000003	0.000021	0.000024
9	0.000006	0.000022	0.000028
10	0.000004	0.000038	0.000042
11	0.000002	0.000024	0.000026
12	0.000006	0.000030	0.000036
13	0.000004	0.000026	0.000030
14	0.000003	0.000023	0.000026
15	0.000005	0.000033	0.000038
16	0.000006	0.000021	0.000027
17	0.000006	0.000020	0.000026
18	0.000006	0.000016	0.000022
19	0.000002	0.000024	0.000026
20	0.000004	0.000021	0.000025
21	0.000004	0.000021	0.000025
22	0.000002	0.000020	0.000022
23	0.000006	0.000016	0.000022
24	0.000003	0.000021	0.000024
25	0.000004	0.000023	0.000027
26	0.000004	0.000022	0.000026
27	0.000005	0.000022	0.000027
28	0.000006	0.000027	0.000033
29	0.000007	0.000022	0.000029
30	0.000003	0.000020	0.000023

b. $N_c = 8, N_l = 1$

Treatment	c_{et}	c_{tt}	c_t
1	0.000007	0.000029	0.000036
2	0.000006	0.000029	0.000035
3	0.000007	0.000027	0.000034
4	0.000009	0.000029	0.000038
5	0.000007	0.000031	0.000038
6	0.000006	0.000026	0.000032
7	0.000005	0.000027	0.000032
8	0.000005	0.000031	0.000036
9	0.000006	0.000028	0.000034
10	0.000007	0.000032	0.000039
11	0.000006	0.000029	0.000035
12	0.000006	0.000029	0.000035
13	0.000006	0.000026	0.000032
14	0.000006	0.000033	0.000039
15	0.000006	0.000026	0.000032
16	0.000003	0.000029	0.000032
17	0.000006	0.000026	0.000032
18	0.000005	0.000027	0.000032
19	0.000004	0.000031	0.000035
20	0.000007	0.000028	0.000035
21	0.000005	0.000025	0.000030
22	0.000005	0.000029	0.000034
23	0.000006	0.000028	0.000034
24	0.000007	0.000029	0.000036
25	0.000007	0.000035	0.000042
26	0.000005	0.000027	0.000032
27	0.000004	0.000030	0.000034
28	0.000004	0.000030	0.000034
29	0.000003	0.000029	0.000032
30	0.000004	0.000028	0.000032

(Cont.)

c. $N_c = 15, N_l = 1$

Treatment	c_{et}	c_{it}	c_t
1	0.000006	0.000040	0.000046
2	0.000009	0.000043	0.000052
3	0.000008	0.000045	0.000053
4	0.000006	0.000031	0.000037
5	0.000006	0.000030	0.000036
6	0.000008	0.000045	0.000053
7	0.000009	0.000037	0.000046
8	0.000010	0.000033	0.000043
9	0.000008	0.000026	0.000034
10	0.000007	0.000024	0.000031
11	0.000005	0.000021	0.000026
12	0.000006	0.000024	0.000030
13	0.000009	0.000038	0.000047
14	0.000004	0.000042	0.000046
15	0.000009	0.000041	0.000050
16	0.000007	0.000048	0.000055
17	0.000010	0.000024	0.000034
18	0.000007	0.000042	0.000049
19	0.000005	0.000042	0.000047
20	0.000006	0.000029	0.000035
21	0.000006	0.000046	0.000052
22	0.000007	0.000031	0.000038
23	0.000006	0.000038	0.000044
24	0.000006	0.000015	0.000021
25	0.000006	0.000027	0.000033
26	0.000010	0.000028	0.000038
27	0.000007	0.000024	0.000031
28	0.000007	0.000034	0.000041
29	0.000008	0.000017	0.000025
30	0.000006	0.000026	0.000032

d. $N_c = 21, N_l = 1$

Treatment	c_{et}	c_{it}	c_t
1	0.000009	0.000032	0.000041
2	0.000007	0.000048	0.000055
3	0.000007	0.000035	0.000042
4	0.000007	0.000047	0.000054
5	0.000006	0.000067	0.000073
6	0.000008	0.000029	0.000037
7	0.000008	0.000021	0.000029
8	0.000009	0.000042	0.000051
9	0.000008	0.000034	0.000042
10	0.000009	0.000038	0.000047
11	0.000010	0.000032	0.000042
12	0.000008	0.000035	0.000043
13	0.000009	0.000038	0.000047
14	0.000009	0.000033	0.000042
15	0.000005	0.000056	0.000061
16	0.000008	0.000033	0.000041
17	0.000009	0.000030	0.000039
18	0.000010	0.000035	0.000045
19	0.000006	0.000047	0.000053
20	0.000006	0.000039	0.000045
21	0.000009	0.000030	0.000039
22	0.000009	0.000034	0.000043
23	0.000010	0.000032	0.000042
24	0.000006	0.000036	0.000042
25	0.000009	0.000036	0.000045
26	0.000007	0.000041	0.000048
27	0.000010	0.000030	0.000040
28	0.000009	0.000049	0.000058
29	0.000006	0.000039	0.000045
30	0.000009	0.000040	0.000049

(Cont.)

e. $N_c = 4, N_l = 2$

Treatment	c_{et}	c_n	c_t
1	0.000012	0.000045	0.000057
2	0.000008	0.000051	0.000059
3	0.000007	0.000035	0.000042
4	0.000008	0.000037	0.000045
5	0.000007	0.000054	0.000061
6	0.000006	0.000047	0.000053
7	0.000006	0.000037	0.000043
8	0.000006	0.000066	0.000072
9	0.000007	0.000040	0.000047
10	0.000007	0.000041	0.000048
11	0.000006	0.000052	0.000058
12	0.000005	0.000046	0.000051
13	0.000006	0.000054	0.000060
14	0.000007	0.000038	0.000045
15	0.000006	0.000043	0.000049
16	0.000009	0.000038	0.000047
17	0.000009	0.000043	0.000052
18	0.000007	0.000052	0.000059
19	0.000008	0.000041	0.000049
20	0.000009	0.000049	0.000058
21	0.000007	0.000045	0.000052
22	0.000008	0.000056	0.000064
23	0.000010	0.000051	0.000061
24	0.000008	0.000044	0.000052
25	0.000008	0.000044	0.000052
26	0.000008	0.000038	0.000046
27	0.000008	0.000048	0.000056
28	0.000008	0.000044	0.000052
29	0.000009	0.000036	0.000045
30	0.000008	0.000054	0.000062

f. $N_c = 8, N_l = 2$

Treatment	c_{et}	c_n	c_t
1	0.000011	0.000063	0.000074
2	0.000010	0.000056	0.000066
3	0.000011	0.000054	0.000065
4	0.000009	0.000057	0.000066
5	0.000010	0.000042	0.000052
6	0.000009	0.000054	0.000063
7	0.000010	0.000056	0.000066
8	0.000011	0.000065	0.000076
9	0.000009	0.000042	0.000051
10	0.000011	0.000051	0.000062
11	0.000009	0.000054	0.000063
12	0.000009	0.000045	0.000054
13	0.000013	0.000070	0.000083
14	0.000010	0.000049	0.000059
15	0.000007	0.000043	0.000050
16	0.000009	0.000046	0.000055
17	0.000009	0.000054	0.000063
18	0.000009	0.000051	0.000060
19	0.000010	0.000054	0.000064
20	0.000009	0.000071	0.000080
21	0.000009	0.000076	0.000085
22	0.000011	0.000049	0.000060
23	0.000011	0.000045	0.000056
24	0.000011	0.000058	0.000069
25	0.000011	0.000051	0.000062
26	0.000009	0.000061	0.000070
27	0.000010	0.000054	0.000064
28	0.000010	0.000056	0.000066
29	0.000009	0.000056	0.000065
30	0.000010	0.000055	0.000065

(Cont.)

g. $N_c = 15, N_t = 2$

Treatment	c_{et}	c_{tt}	c_t
1	0.000009	0.000060	0.000069
2	0.000017	0.000048	0.000065
3	0.000012	0.000060	0.000072
4	0.000010	0.000067	0.000077
5	0.000018	0.000053	0.000071
6	0.000017	0.000072	0.000089
7	0.000022	0.000062	0.000084
8	0.000009	0.000051	0.000060
9	0.000011	0.000069	0.000080
10	0.000011	0.000052	0.000063
11	0.000010	0.000051	0.000061
12	0.000017	0.000048	0.000065
13	0.000012	0.000068	0.000080
14	0.000008	0.000039	0.000047
15	0.000012	0.000048	0.000060
16	0.000018	0.000067	0.000085
17	0.000012	0.000052	0.000064
18	0.000020	0.000049	0.000069
19	0.000011	0.000071	0.000082
20	0.000022	0.000051	0.000073
21	0.000018	0.000064	0.000082
22	0.000013	0.000051	0.000064
23	0.000013	0.000059	0.000072
24	0.000010	0.000060	0.000070
25	0.000018	0.000056	0.000074
26	0.000014	0.000055	0.000069
27	0.000010	0.000072	0.000082
28	0.000016	0.000066	0.000082
29	0.000017	0.000060	0.000077
30	0.000010	0.000074	0.000084

h. $N_c = 21, N_t = 2$

Treatment	c_{et}	c_{tt}	c_t
1	0.000019	0.000053	0.000072
2	0.000020	0.000054	0.000074
3	0.000026	0.000066	0.000092
4	0.000017	0.000071	0.000088
5	0.000020	0.000070	0.000090
6	0.000014	0.000063	0.000077
7	0.000021	0.000046	0.000067
8	0.000016	0.000076	0.000092
9	0.000021	0.000053	0.000074
10	0.000010	0.000055	0.000065
11	0.000013	0.000083	0.000096
12	0.000023	0.000050	0.000073
13	0.000019	0.000066	0.000085
14	0.000018	0.000040	0.000058
15	0.000014	0.000051	0.000065
16	0.000011	0.000069	0.000080
17	0.000020	0.000058	0.000078
18	0.000018	0.000045	0.000063
19	0.000019	0.000061	0.000080
20	0.000014	0.000073	0.000087
21	0.000020	0.000048	0.000068
22	0.000020	0.000078	0.000098
23	0.000022	0.000075	0.000097
24	0.000020	0.000063	0.000083
25	0.000020	0.000047	0.000067
26	0.000018	0.000054	0.000072
27	0.000025	0.000062	0.000087
28	0.000019	0.000069	0.000088
29	0.000018	0.000064	0.000082
30	0.000020	0.000057	0.000077

(Cont.)

i. $N_c = 4, N_l = 3$

Treatment	c_{ct}	c_{tt}	c_t
1	0.000011	0.000051	0.000062
2	0.000010	0.000056	0.000066
3	0.000011	0.000070	0.000081
4	0.000007	0.000070	0.000077
5	0.000011	0.000057	0.000068
6	0.000014	0.000078	0.000092
7	0.000010	0.000081	0.000091
8	0.000011	0.000061	0.000072
9	0.000011	0.000054	0.000065
10	0.000009	0.000056	0.000065
11	0.000011	0.000055	0.000066
12	0.000009	0.000059	0.000068
13	0.000011	0.000059	0.000070
14	0.000009	0.000074	0.000083
15	0.000013	0.000053	0.000066
16	0.000010	0.000045	0.000055
17	0.000014	0.000058	0.000072
18	0.000010	0.000054	0.000064
19	0.000009	0.000084	0.000093
20	0.000012	0.000071	0.000083
21	0.000010	0.000050	0.000060
22	0.000009	0.000080	0.000089
23	0.000011	0.000053	0.000064
24	0.000012	0.000072	0.000084
25	0.000011	0.000053	0.000064
26	0.000012	0.000049	0.000061
27	0.000011	0.000059	0.000070
28	0.000009	0.000068	0.000077
29	0.000010	0.000055	0.000065
30	0.000012	0.000061	0.000073

j. $N_c = 8, N_l = 3$

Treatment	c_{ct}	c_{tt}	c_t
1	0.000011	0.000044	0.000055
2	0.000016	0.000056	0.000072
3	0.000012	0.000055	0.000067
4	0.000012	0.000052	0.000064
5	0.000010	0.000048	0.000058
6	0.000014	0.000058	0.000072
7	0.000014	0.000074	0.000088
8	0.000014	0.000066	0.000080
9	0.000011	0.000045	0.000056
10	0.000013	0.000048	0.000061
11	0.000012	0.000056	0.000068
12	0.000014	0.000053	0.000067
13	0.000014	0.000038	0.000052
14	0.000014	0.000062	0.000076
15	0.000014	0.000065	0.000079
16	0.000014	0.000079	0.000093
17	0.000013	0.000079	0.000092
18	0.000014	0.000064	0.000078
19	0.000016	0.000064	0.000080
20	0.000017	0.000050	0.000067
21	0.000016	0.000052	0.000068
22	0.000017	0.000059	0.000076
23	0.000014	0.000046	0.000060
24	0.000011	0.000065	0.000076
25	0.000010	0.000052	0.000062
26	0.000016	0.000052	0.000068
27	0.000016	0.000073	0.000089
28	0.000014	0.000052	0.000066
29	0.000014	0.000058	0.000072
30	0.000015	0.000075	0.000090

(Cont.)

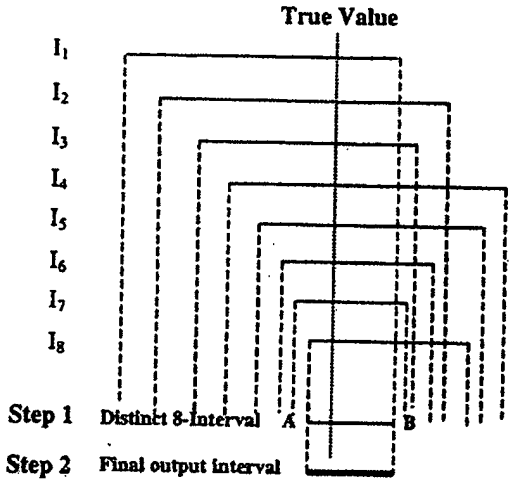
k. $N_c = 15, N_l = 3$

Treatment	c_{et}	c_{it}	c_t
1	0.000018	0.000085	0.000103
2	0.000013	0.000060	0.000073
3	0.000012	0.000076	0.000088
4	0.000018	0.000066	0.000084
5	0.000012	0.000061	0.000073
6	0.000014	0.000082	0.000096
7	0.000023	0.000076	0.000099
8	0.000019	0.000060	0.000079
9	0.000014	0.000077	0.000091
10	0.000017	0.000058	0.000075
11	0.000016	0.000039	0.000055
12	0.000018	0.000078	0.000096
13	0.000013	0.000066	0.000079
14	0.000026	0.000081	0.000107
15	0.000022	0.000062	0.000084
16	0.000011	0.000068	0.000079
17	0.000014	0.000062	0.000076
18	0.000013	0.000067	0.000080
19	0.000014	0.000048	0.000062
20	0.000022	0.000077	0.000099
21	0.000014	0.000083	0.000097
22	0.000014	0.000055	0.000069
23	0.000021	0.000080	0.000101
24	0.000025	0.000050	0.000075
25	0.000015	0.000060	0.000075
26	0.000016	0.000067	0.000083
27	0.000018	0.000049	0.000067
28	0.000026	0.000077	0.000103
29	0.000014	0.000084	0.000098
30	0.000016	0.000059	0.000075

l. $N_c = 21, N_l = 3$

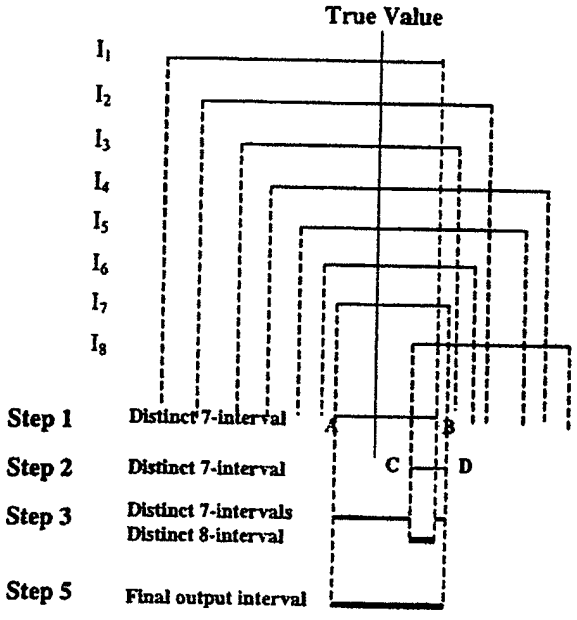
Treatment	c_{et}	c_{it}	c_t
1	0.000018	0.000083	0.000101
2	0.000022	0.000069	0.000091
3	0.000018	0.000073	0.000091
4	0.000022	0.000077	0.000099
5	0.000023	0.000083	0.000106
6	0.000020	0.000051	0.000071
7	0.000022	0.000092	0.000114
8	0.000026	0.000039	0.000065
9	0.000019	0.000147	0.000166
10	0.000030	0.000070	0.000100
11	0.000023	0.000085	0.000108
12	0.000021	0.000044	0.000065
13	0.000017	0.000076	0.000093
14	0.000022	0.000085	0.000107
15	0.000026	0.000055	0.000081
16	0.000026	0.000063	0.000089
17	0.000019	0.000045	0.000064
18	0.000022	0.000056	0.000078
19	0.000028	0.000063	0.000091
20	0.000026	0.000067	0.000093
21	0.000036	0.000060	0.000096
22	0.000025	0.000068	0.000093
23	0.000024	0.000076	0.000100
24	0.000024	0.000054	0.000078
25	0.000020	0.000063	0.000083
26	0.000028	0.000049	0.000077
27	0.000022	0.000047	0.000069
28	0.000029	0.000056	0.000085
29	0.000022	0.000051	0.000073
30	0.000024	0.000070	0.000094

APPENDIX B Examples of Execution of FTSIA



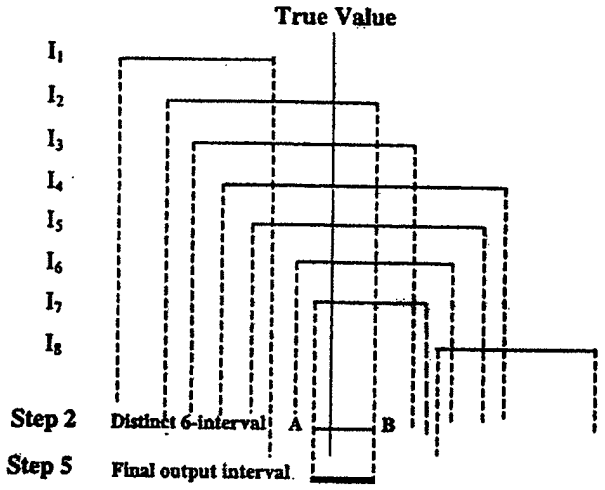
Steps:
N_s-f=8
Step 1. Find out distinct 8-interval from abstract sensors I_1 to I_8 AB
Step 2. Since there is no other abstract sensors, AB is the final output interval

Figure B.1 Example of Execution of FTSIA When $N_s = 8, f = 0$



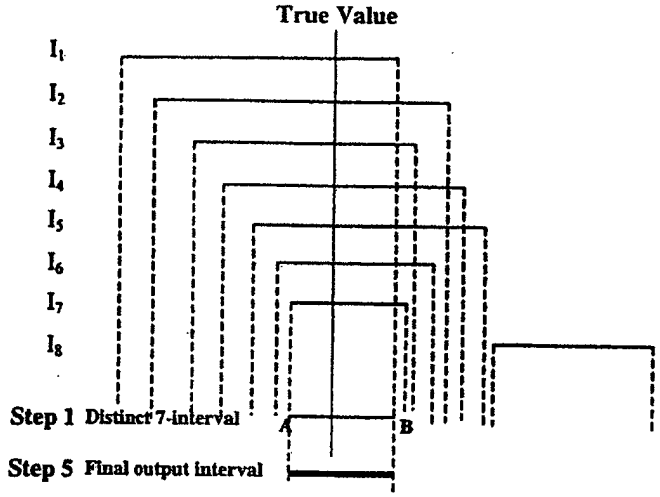
Steps:
N_s-f=7
Step 1. Find out the distinct 7-Interval AB from first seven abstract sensors AB
Step 2. Ignore I_1 because its higher bound is the smallest among the first seven abstract sensors and find out the distinct 7-interval from abstract sensors I_2 to I_8 CD
Step 3. Combine distinct intervals and classify different levels of distinct intervals (AC and BD are distinct 7-intervals; CB is a distinct 8-interval)
Step 4. Compare the distinct intervals with all the abstract sensors and find out the possibly faulty sensors: if AC contains the true value, I_8 is tamely faulty; if CD contains the true value, all abstract sensors are correct; if BC contains the true value, I_1 is tamely faulty. Therefore, I_8 and I_1 are possibly tamely faulty.
Step 5. Ignore readings from all the possibly tamely faulty sensors, and find out final output interval from the rest correct sensors.

Figure B.2 Example of Execution of FTSIA When $N_s = 8, f_t = 1$



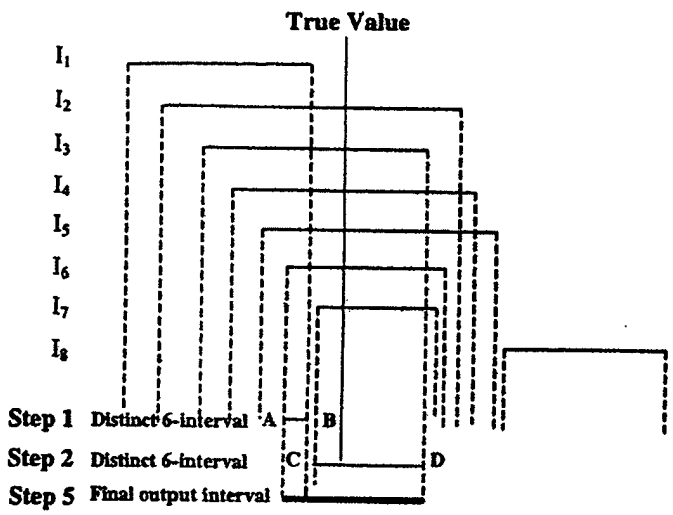
Steps:
N_s-f=6
Step 1. There is no distinct 6-Interval from first six abstract sensors
Step 2. Ignore I_1 because its higher bound is the smallest among the first six abstract sensors and find out the distinct 6-interval from abstract sensors I_2 to I_7 AB
Step 3. Ignore reading from I_2, and there is no distinct 6-interval from abstract I_3 sensors to I_8
Step 4. Compare the distinct intervals with all the abstract sensors and find out the possibly faulty sensors. Since AB is the only distinct 6-interval, I_1 and I_8 must be faulty.
Step 5. AB is the final output interval.

Figure B.3 Example of Execution of FTSIA When $N_s = 8, f_t = 2$



Steps:
N_s-f=7
Step 1. Find out the distinct 7-interval from abstract sensors I_1 to I_7 AB
Step 2. Ignore I_1 because its higher bound is the smallest among the first seven abstract sensors and there is no distinct 7-interval from abstract sensors I_2 to I_8
Step 3. Compare the distinct intervals with all the abstract sensors and find out the possibly faulty sensors. Since AB is only distinct 7-interval, I_8 must be faulty
Step 4. AB is the final output interval.

Figure B.4 Example of Execution of FTSIA When $N_s = 8, f_w = 1$



Steps:
N_s-f=6
Step 1. Find out the distinct 6-interval from abstract sensors I_1 to I_6 AB
Step 2. Ignore I_1 because its higher bound is the smallest among the first six abstract sensors and find the distinct 6-interval from abstract sensors I_2 to I_7 CD
Step 3. Combine distinct intervals and classify different levels of distinct intervals (both AC and BD are distinct 6-intervals)
Step 4. Compare the distinct intervals with all the abstract sensors and find out the possibly faulty sensors: if AB contains the true value, I_7 to I_8 are faulty; if CD contains the true value, I_1 to I_8 are faulty. Therefore, I_1 and I_7 are possibly tamely faulty, and I_8 must be faulty.
Step 5. Ignore readings from possibly faulty and faulty sensors, find out the final output interval from the rest correct sensors

Figure B.5 Example of Execution of FTSIA When $N_s = 8, f_t = 1, f_w = 1$

APPENDIX C Experiment Results of FTSIA and Statistical Calculations (τ^*=0.5, ε=±0.005)

In the statistical calculations, the formula used for calculating mean is $\frac{1}{N_s}(\sum_{i=1}^{N_s}\tau_i)$ and the formula used for calculating standard deviation (Std.) is $\sqrt{\frac{N_s\sum_{i=1}^{N_s}\tau_i^2-(\sum_{i=1}^{N_s}\tau_i)^2}{N_s(N_s-1)}}$

Table C.1 Experiment Results of FTSIA and Statistical Calculations (Case 1)

a. $N_s = 8, f = 0$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49796252	0.50599591	0.5019792	0.0019792	0.5020511	0.0020511	0.0007325
2	0.49979105	0.50508982	0.5024404	0.0024404	0.5020586	0.0020586	0.0018668
3	0.49926955	0.50551097	0.5023903	0.0023903	0.5024965	0.0024965	0.0012247
4	0.49980358	0.50579084	0.5027972	0.0027972	0.5029328	0.0029328	0.0014704
5	0.49995181	0.50518419	0.502568	0.002568	0.5021969	0.0021969	0.0015634
6	0.49812857	0.50567376	0.5019012	0.0019012	0.5021544	0.0021544	0.0008269
7	0.4999855	0.50513723	0.5025614	0.0025614	0.5030277	0.0030277	0.0015744
8	0.49980421	0.50565663	0.5027304	0.0027304	0.5026188	0.0026188	0.0011699
9	0.49987774	0.50538874	0.5026332	0.0026332	0.5022356	0.0022356	0.0017834
10	0.49960093	0.50522843	0.5024147	0.0024147	0.5023244	0.0023244	0.0014075
11	0.49991374	0.50523443	0.5025741	0.0025741	0.5030171	0.0030171	0.0014693
12	0.49874722	0.5050154	0.5018813	0.0018813	0.5022298	0.0022298	0.0012337
Average	0.49940304	0.505408863	0.502406	0.002406	0.5024453	0.0024453	

b. $N_s = 15, f = 0$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49928084	0.50534114	0.502311	0.002311	0.5022704	0.0022704	0.0009459
2	0.49910186	0.50500907	0.5020555	0.0020555	0.5019179	0.0019179	0.0012485
3	0.49954836	0.5053165	0.5024324	0.0024324	0.5030002	0.0030002	0.0014989
4	0.49939947	0.50512739	0.5022634	0.0022634	0.5022939	0.0022939	0.0015699
5	0.49936191	0.50543349	0.5023977	0.0023977	0.5028966	0.0028966	0.0013283
6	0.49995425	0.50508228	0.5025183	0.0025183	0.5022597	0.0022597	0.0014881
7	0.4996912	0.50504013	0.5023657	0.0023657	0.5027383	0.0027383	0.001482
8	0.49930976	0.50525148	0.5022806	0.0022806	0.5023218	0.0023218	0.0012336
9	0.49980147	0.50599519	0.5028983	0.0028983	0.5029607	0.0029607	0.0011435
10	0.49978751	0.50534796	0.5025677	0.0025677	0.502793	0.002793	0.0012274
11	0.49958979	0.50560502	0.5025974	0.0025974	0.5026188	0.0026188	0.0012875
12	0.49992005	0.50501645	0.5024683	0.0024683	0.5026511	0.0026511	0.0016061
Average	0.49956221	0.505297175	0.5024297	0.0024297	0.5025602	0.0025602	

(Cont.)

c. $N_s = 21, f = 0$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49940835	0.50521986	0.5023141	0.0023141	0.5023434	0.0023434	0.001434
2	0.49991704	0.50527108	0.5025941	0.0025941	0.5025089	0.0025089	0.0015061
3	0.49951541	0.50501712	0.5022663	0.0022663	0.5024916	0.0024916	0.0013459
4	0.49997988	0.50540742	0.5026937	0.0026937	0.5029658	0.0029658	0.0015172
5	0.49980581	0.50500611	0.502406	0.002406	0.5021951	0.0021951	0.0016855
6	0.49947866	0.50550706	0.5024929	0.0024929	0.502355	0.002355	0.001274
7	0.49981117	0.50501519	0.5024132	0.0024132	0.5022662	0.0022662	0.0015969
8	0.4998184	0.50530213	0.5025603	0.0025603	0.5025368	0.0025368	0.0012457
9	0.49898706	0.5052208	0.5021039	0.0021039	0.5022881	0.0022881	0.0011116
10	0.49967244	0.50510212	0.5023873	0.0023873	0.5025692	0.0025692	0.0015299
11	0.49940494	0.50500673	0.5022058	0.0022058	0.5022872	0.0022872	0.0012085
12	0.49998991	0.50524911	0.5026195	0.0026195	0.5029167	0.0029167	0.0015229
Average	0.49964909	0.505193728	0.5024214	0.0024214	0.502477	0.002477	

d. $N_s = 27, f = 0$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49983935	0.50505835	0.5024489	0.0024489	0.5023792	0.0023792	0.0013202
2	0.49962378	0.50526936	0.5024466	0.0024466	0.5024579	0.0024579	0.0013043
3	0.49933206	0.50502746	0.5021798	0.0021798	0.5020785	0.0020785	0.001371
4	0.49998649	0.5050075	0.502497	0.002497	0.5026989	0.0026989	0.0014821
5	0.49971876	0.50512297	0.5024209	0.0024209	0.5023309	0.0023309	0.0011845
6	0.49996842	0.5050165	0.5024925	0.0024925	0.5024155	0.0024155	0.0016019
7	0.49995932	0.50506687	0.5025131	0.0025131	0.5029196	0.0029196	0.0014238
8	0.49998378	0.50516355	0.5025737	0.0025737	0.5025107	0.0025107	0.0015578
9	0.49995396	0.5050368	0.5024954	0.0024954	0.502648	0.002648	0.0014875
10	0.49976186	0.50504613	0.502404	0.002404	0.5023729	0.0023729	0.0015783
11	0.49979355	0.50517038	0.502482	0.002482	0.5027683	0.0027683	0.0014278
12	0.49958318	0.50504776	0.5023155	0.0023155	0.502562	0.002562	0.0013961
Average	0.49979204	0.505086136	0.5024391	0.0024391	0.5025119	0.0025119	

e. $N_s = 36, f = 0$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49981836	0.50506372	0.502441	0.002441	0.5023291	0.0023291	0.0013947
2	0.49993868	0.50506525	0.502502	0.002502	0.5025447	0.0025447	0.0014281
3	0.49992337	0.50508638	0.5025049	0.0025049	0.5023862	0.0023862	0.0014936
4	0.49995917	0.5050574	0.5025083	0.0025083	0.5028427	0.0028427	0.0015979
5	0.49992327	0.50500978	0.5024665	0.0024665	0.5026589	0.0026589	0.001589
6	0.49982858	0.50513719	0.5024829	0.0024829	0.5023423	0.0023423	0.0014108
7	0.49990614	0.50525254	0.5025793	0.0025793	0.5025357	0.0025357	0.0013777
8	0.49987083	0.50517544	0.5025231	0.0025231	0.5022054	0.0022054	0.0015986
9	0.49983627	0.50563913	0.5027377	0.0027377	0.5025759	0.0025759	0.0012234
10	0.4998221	0.5052905	0.5025563	0.0025563	0.5028588	0.0028588	0.0013412
11	0.49996308	0.5051605	0.5025618	0.0025618	0.5025942	0.0025942	0.0014594
12	0.49987818	0.50517756	0.5025279	0.0025279	0.5026436	0.0026436	0.001426
Average	0.499889	0.505176283	0.5025326	0.0025326	0.5025431	0.0025431	

Table C.2 Experiment Results of FTSIA and Statistical Calculations (Case 2)

a. $N_s = 8, f_t = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49936145	0.5065619	0.5029617	0.0029617	0.5038056	0.0038056	0.0027807
2	0.49934696	0.50627515	0.5028111	0.0028111	0.503102	0.003102	0.0022272
3	0.4989259	0.50669498	0.5028104	0.0028104	0.5032258	0.0032258	0.0024381
4	0.49747933	0.50552288	0.5015011	0.0015011	0.5017775	0.0017775	0.0018819
5	0.497396	0.50590588	0.5016509	0.0016509	0.5022772	0.0022772	0.0022951
6	0.49807041	0.50541893	0.5017447	0.0017447	0.5020745	0.0020745	0.0023422
7	0.49977289	0.50549349	0.5026332	0.0026332	0.5025069	0.0025069	0.002762
8	0.49933211	0.50597289	0.5026525	0.0026525	0.5030647	0.0030647	0.0030047
9	0.49952402	0.50649848	0.5030113	0.0030113	0.5033534	0.0033534	0.0020561
10	0.49864137	0.50733809	0.5029897	0.0029897	0.5033138	0.0033138	0.0020507
11	0.49894644	0.50695258	0.5029495	0.0029495	0.5029276	0.0029276	0.0020002
12	0.49914953	0.50578568	0.5024676	0.0024676	0.5032767	0.0032767	0.0029671
Average	0.49882887	0.506201744	0.5025153	0.0025153	0.5028921	0.0028921	

b. $N_s = 8, f_t = 2$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49914953	0.50679545	0.5029725	0.0029725	0.5039017	0.0039017	0.0035406
2	0.49951886	0.50719015	0.5033545	0.0033545	0.5036211	0.0036211	0.0026768
3	0.49991709	0.5064845	0.5032008	0.0032008	0.5039179	0.0039179	0.0032475
4	0.49927258	0.50763448	0.5034535	0.0034535	0.5039449	0.0039449	0.0022883
5	0.49886323	0.50583714	0.5023502	0.0023502	0.5030506	0.0030506	0.0027044
6	0.49858124	0.50610477	0.502343	0.002343	0.5035919	0.0035919	0.0031037
7	0.49912138	0.50636803	0.5027447	0.0027447	0.5033371	0.0033371	0.0030249
8	0.49856593	0.50667595	0.5026209	0.0026209	0.5033062	0.0033062	0.0032311
9	0.49770695	0.50635045	0.5020287	0.0020287	0.5032618	0.0032618	0.003604
10	0.49798286	0.50603726	0.5020101	0.0020101	0.50337	0.00337	0.0033615
11	0.49966803	0.50701709	0.5033426	0.0033426	0.5038428	0.0038428	0.0028248
12	0.49948746	0.50736066	0.5034241	0.0034241	0.5038157	0.0038157	0.0032612
Average	0.49898626	0.506654661	0.5028205	0.0028205	0.5035801	0.0035801	

(Cont.)

c. $N_s = 8, f_t = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49891307	0.50824385	0.5035785	0.0035785	0.504157	0.004157	0.0032255
2	0.49918214	0.50884185	0.504012	0.004012	0.504896	0.004896	0.0032092
3	0.49981912	0.5096614	0.5047403	0.0047403	0.5058456	0.0058456	0.0026614
4	0.49703591	0.50699213	0.502014	0.002014	0.5037965	0.0037965	0.0036467
5	0.4975694	0.50710928	0.5023393	0.0023393	0.5033403	0.0033403	0.0028803
6	0.49897776	0.50863352	0.5038056	0.0038056	0.5043195	0.0043195	0.0034355
7	0.4976622	0.5075267	0.5025945	0.0025945	0.5043287	0.0043287	0.0034448
8	0.49901937	0.50787969	0.5034495	0.0034495	0.5043575	0.0043575	0.0029468
9	0.49961377	0.50791225	0.503763	0.003763	0.5044677	0.0044677	0.0029117
10	0.49885962	0.50877049	0.5038151	0.0038151	0.5045478	0.0045478	0.0029251
11	0.49999813	0.50798293	0.5039905	0.0039905	0.5048647	0.0048647	0.0033054
12	0.49895114	0.50727876	0.503115	0.003115	0.5039462	0.0039462	0.0032327
Average	0.49880014	0.508069404	0.5034348	0.0034348	0.5044056	0.0044056	

d. $N_s = 15, f_t = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49909356	0.50572427	0.5024089	0.0024089	0.5026127	0.0026127	0.0022849
2	0.49932539	0.50542086	0.5023731	0.0023731	0.5026977	0.0026977	0.0021969
3	0.4996285	0.5055161	0.5025723	0.0025723	0.5028187	0.0028187	0.0018977
4	0.4995017	0.50541406	0.5024579	0.0024579	0.502484	0.002484	0.002241
5	0.49971311	0.505589	0.5026511	0.0026511	0.5028167	0.0028167	0.0023155
6	0.49950075	0.50542465	0.5024627	0.0024627	0.5025368	0.0025368	0.0020929
7	0.49865443	0.50570372	0.5021791	0.0021791	0.5025509	0.0025509	0.0017688
8	0.49991073	0.50520296	0.5025569	0.0025568	0.5030292	0.0030292	0.0026083
9	0.49959115	0.50573746	0.5026643	0.0026643	0.5028845	0.0028845	0.0020643
10	0.49980359	0.50537178	0.5025877	0.0025877	0.5025324	0.0025324	0.0021395
11	0.49984095	0.50575773	0.5027993	0.0027993	0.5029165	0.0029165	0.0024803
12	0.49986476	0.50569074	0.5027778	0.0027778	0.5029558	0.0029558	0.001957
Average	0.49953572	0.505546111	0.5025409	0.0025409	0.5027363	0.0027363	

e. $N_s = 15, f_t = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49970868	0.50682646	0.5032676	0.0032676	0.5035006	0.0035006	0.0026801
2	0.49942348	0.50629661	0.5028601	0.0028601	0.5034681	0.0034681	0.0026445
3	0.49976561	0.50742117	0.5035934	0.0035934	0.5041834	0.0041834	0.0024974
4	0.49888065	0.50723518	0.5030579	0.0030579	0.5033432	0.0033432	0.0018143
5	0.49882218	0.50593075	0.5023765	0.0023765	0.5032316	0.0032316	0.0024873
6	0.49937553	0.50575139	0.5025635	0.0025635	0.5033534	0.0033534	0.0028113
7	0.49982171	0.50657344	0.5031976	0.0031976	0.5033475	0.0033475	0.0024491
8	0.4995684	0.50616925	0.5028688	0.0028688	0.5035307	0.0035307	0.0029544
9	0.49942638	0.50642014	0.5029233	0.0029233	0.5032869	0.0032869	0.0026499
10	0.49959378	0.50661158	0.5031027	0.0031027	0.5036319	0.0036319	0.0022404
11	0.49970677	0.50621373	0.5029603	0.0029603	0.5034017	0.0034017	0.00236
12	0.49945674	0.50623522	0.502846	0.002846	0.5032195	0.0032195	0.0026946
Average	0.49946249	0.506473743	0.5029681	0.0029681	0.5034582	0.0034582	

(Cont.)

f. $N_s = 15, f_t = 5$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49987083	0.50746842	0.5036696	0.0036696	0.5043337	0.0043337	0.003777
2	0.4993184	0.50848492	0.5039017	0.0039017	0.5043518	0.0043518	0.0028738
3	0.49963695	0.50687051	0.5032537	0.0032537	0.5043026	0.0043026	0.0035135
4	0.49941426	0.50755314	0.5034837	0.0034837	0.5039954	0.0039954	0.0027635
5	0.4995663	0.50722234	0.5033943	0.0033943	0.5044185	0.0044185	0.0028373
6	0.49984025	0.50710237	0.5034713	0.0034713	0.5042335	0.0042335	0.0032507
7	0.49998819	0.50636643	0.5031773	0.0031773	0.5041108	0.0041108	0.0034147
8	0.49974628	0.50749325	0.5036198	0.0036198	0.5039659	0.0039659	0.0022068
9	0.49919771	0.50736576	0.5032817	0.0032817	0.504167	0.004167	0.0029304
10	0.49998293	0.5083734	0.5041782	0.0041782	0.5046221	0.0046221	0.0028168
11	0.49980767	0.50705643	0.5034321	0.0034321	0.504032	0.004032	0.002995
12	0.49928838	0.50718763	0.503238	0.003238	0.5044149	0.0044149	0.003177
Average	0.49963818	0.507378717	0.5035085	0.0035085	0.5042457	0.0042457	

g. $N_s = 15, f_t = 7$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49908182	0.50908182	0.5040818	0.0040818	0.5046128	0.0046128	0.0035794
2	0.49880045	0.50880045	0.5038005	0.0038005	0.5042373	0.0042373	0.0029527
3	0.49890159	0.50890159	0.5039016	0.0039016	0.5055221	0.0055221	0.0030501
4	0.49854646	0.50854647	0.5035465	0.0035465	0.5043975	0.0043975	0.0035102
5	0.4992234	0.5092234	0.5042234	0.0042234	0.5052727	0.0052727	0.0026531
6	0.49899324	0.50899324	0.5039932	0.0039932	0.5045131	0.0045131	0.0033109
7	0.49931998	0.50931998	0.50432	0.00432	0.5050514	0.0050514	0.0029778
8	0.49892765	0.50892765	0.5039277	0.0039277	0.5051088	0.0051088	0.0030651
9	0.49942244	0.50942244	0.5044224	0.0044224	0.5049713	0.0049713	0.0030557
10	0.49986701	0.50986701	0.504867	0.004867	0.5053539	0.0053539	0.0029991
11	0.49936981	0.50936981	0.5043698	0.0043698	0.5051178	0.0051178	0.0029175
12	0.49950319	0.50950319	0.5045032	0.0045032	0.5059093	0.0059093	0.0030103
Average	0.49916309	0.509163088	0.5041631	0.0041631	0.5050057	0.0050057	

h. $N_s = 21, f_t = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49920187	0.5050874	0.5021446	0.0021446	0.5021843	0.0021843	0.0015967
2	0.49907606	0.50541744	0.5022468	0.0022468	0.5025643	0.0025643	0.0018831
3	0.49967122	0.50554719	0.5026092	0.0026092	0.5027374	0.0027374	0.0020205
4	0.49967618	0.50569237	0.5026843	0.0026843	0.5027957	0.0027957	0.0019476
5	0.49978336	0.50503005	0.5024067	0.0024067	0.5024649	0.0024649	0.0020494
6	0.49944217	0.505446	0.5024441	0.0024441	0.502667	0.002667	0.0016343
7	0.49978497	0.5058003	0.5027926	0.0027926	0.5028382	0.0028382	0.0020723
8	0.49990121	0.50521825	0.5025597	0.0025597	0.5027188	0.0027188	0.0017539
9	0.49995088	0.50534043	0.5026457	0.0026457	0.5028989	0.0028989	0.0019951
10	0.49989302	0.505159	0.502526	0.002526	0.5024061	0.0024061	0.0018107
11	0.49987116	0.50503286	0.502452	0.002452	0.5029285	0.0029285	0.0019622
12	0.49969013	0.50592841	0.5028093	0.0028093	0.5028085	0.0028085	0.0019928
Average	0.49966185	0.505391642	0.5025268	0.0025268	0.5026677	0.0026677	

(Cont.)

i. $N_s = 21, f_t = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.4996278	0.50539909	0.5025135	0.0025135	0.5026392	0.0026392	0.0025234
2	0.49946925	0.50674555	0.5031074	0.0031074	0.5033072	0.0033072	0.0022555
3	0.4999579	0.50581314	0.5028855	0.0028855	0.5030096	0.0030096	0.0023899
4	0.49912807	0.50530749	0.5022178	0.0022178	0.5026453	0.0026453	0.0027413
5	0.49938447	0.50596042	0.5026725	0.0026724	0.5029878	0.0029878	0.0021412
6	0.499651	0.50544031	0.5025457	0.0025457	0.5029696	0.0029696	0.0024379
7	0.49978823	0.50629452	0.5030414	0.0030414	0.5034586	0.0034586	0.0024426
8	0.49914315	0.50565307	0.5023981	0.0023981	0.5026251	0.0026251	0.0022132
9	0.49925369	0.50548495	0.5023693	0.0023693	0.5027015	0.0027015	0.0023041
10	0.49986755	0.50601061	0.5029391	0.0029391	0.5030544	0.0030544	0.0022048
11	0.49947339	0.50626276	0.5028681	0.0028681	0.5029809	0.0029809	0.0021993
12	0.49918695	0.50567518	0.5024311	0.0024311	0.5024963	0.0024963	0.0021883
Average	0.49949429	0.505837258	0.5026658	0.0026658	0.5029063	0.0029063	

j. $N_s = 21, f_t = 5$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49989728	0.50573381	0.5028156	0.0028156	0.5037183	0.0037183	0.0027985
2	0.49923914	0.50620662	0.5027229	0.0027229	0.503336	0.003336	0.0029422
3	0.49909696	0.50669485	0.5028959	0.0028959	0.5038178	0.0038178	0.0030921
4	0.49931139	0.50651872	0.5029151	0.0029151	0.5036555	0.0036555	0.0023377
5	0.49904724	0.50688365	0.5029655	0.0029655	0.5035252	0.0035252	0.0023126
6	0.49866755	0.50642247	0.502545	0.002545	0.5035636	0.0035636	0.0028635
7	0.49937126	0.50651502	0.5029431	0.0029431	0.5035302	0.0035302	0.0027819
8	0.49891775	0.50678096	0.5028494	0.0028494	0.5033285	0.0033285	0.0027699
9	0.4991188	0.50583317	0.502476	0.002476	0.5030478	0.0030478	0.002568
10	0.49939535	0.50577655	0.502586	0.002586	0.5034675	0.0034675	0.0031506
11	0.49930806	0.50609451	0.5027013	0.0027013	0.5032721	0.0032721	0.0025768
12	0.49987071	0.50631505	0.5030929	0.0030929	0.5035321	0.0035321	0.0027859
Average	0.49927012	0.506314615	0.5027924	0.0027924	0.5034829	0.0034829	

k. $N_s = 21, f_t = 7$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49981846	0.50787374	0.5038461	0.0038461	0.5041158	0.0041158	0.0028689
2	0.49981248	0.50843261	0.5041226	0.0041225	0.5046395	0.0046395	0.0027877
3	0.49991319	0.50794918	0.5039312	0.0039312	0.504658	0.004658	0.0030199
4	0.49980047	0.50826195	0.5040312	0.0040312	0.5043475	0.0043475	0.0024763
5	0.49942243	0.50777226	0.5035974	0.0035973	0.5041457	0.0041457	0.0029134
6	0.49939799	0.50814742	0.5037727	0.0037727	0.5045666	0.0045666	0.0027338
7	0.4988308	0.50750597	0.5031684	0.0031684	0.5043382	0.0043382	0.0033891
8	0.49930107	0.50811252	0.5037068	0.0037068	0.50425	0.00425	0.0027992
9	0.49905259	0.50728429	0.5031684	0.0031684	0.503979	0.003979	0.0028866
10	0.49998888	0.50734449	0.5036667	0.0036667	0.5043319	0.0043319	0.0030015
11	0.49990333	0.50706028	0.5034818	0.0034818	0.5043224	0.0043224	0.0033459
12	0.4991441	0.50753506	0.5033396	0.0033396	0.5040821	0.0040821	0.0029881
Average	0.49953215	0.507773314	0.5036527	0.0036527	0.5043147	0.0043147	

(Cont.) 1. $N_s = 21, f_s = 10$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49968267	0.50968267	0.5046827	0.0046827	0.5051505	0.0051505	0.002984
2	0.49937428	0.50937428	0.5043743	0.0043743	0.5056529	0.0056529	0.0030123
3	0.49937428	0.50937428	0.5043743	0.0043743	0.5052322	0.0052322	0.0027451
4	0.49988151	0.50988151	0.5048815	0.0048815	0.5055707	0.0055707	0.0028647
5	0.49876603	0.50876603	0.503766	0.003766	0.5047361	0.0047361	0.0025604
6	0.49889554	0.50889554	0.5038955	0.0038955	0.5049002	0.0049002	0.003453
7	0.49950986	0.50950986	0.5045099	0.0045099	0.5048014	0.0048014	0.003317
8	0.49948509	0.50948509	0.5044851	0.0044851	0.5049487	0.0049487	0.0032447
9	0.49922582	0.50922582	0.5042258	0.0042258	0.5049345	0.0049345	0.0031369
10	0.49947976	0.50947976	0.5044798	0.0044798	0.5048989	0.0048989	0.0028323
11	0.49933787	0.50933787	0.5043379	0.0043379	0.5051951	0.0051951	0.0032867
12	0.49953019	0.50953019	0.5045302	0.0045302	0.505419	0.005419	0.0029089
Average	0.49937858	0.509378575	0.5043786	0.0043786	0.50512	0.00512	

Table C.3 Experiment Results of FTSIA and Statistical Calculations (Case 3)

a. $N_s = 8, f_w = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49830274	0.50525146	0.5017771	0.0017771	0.5075333	0.0075333	0.016518
2	0.49987353	0.50501107	0.5024423	0.0024423	0.5080563	0.0080563	0.0163823
3	0.49945923	0.50513215	0.5022957	0.0022957	0.507739	0.007739	0.0134797
4	0.49842174	0.50540274	0.5019122	0.0019122	0.5058399	0.0058399	0.0123811
5	0.49915461	0.50562071	0.5023877	0.0023877	0.5069586	0.0069586	0.0121969
6	0.49909087	0.50560486	0.5023479	0.0023479	0.5055159	0.0055159	0.0099107
7	0.49924392	0.50507159	0.5021578	0.0021578	0.5060495	0.0060495	0.0095769
8	0.49881164	0.5052022	0.5020069	0.0020069	0.5049012	0.0049012	0.0074582
9	0.49980127	0.50507573	0.5024385	0.0024385	0.5050885	0.0050885	0.0065384
10	0.49921703	0.50522586	0.5022215	0.0022214	0.5070597	0.0070597	0.0137379
11	0.49977707	0.50503031	0.5024037	0.0024037	0.507041	0.007041	0.013193
12	0.49901493	0.5067047	0.5028598	0.0028598	0.5061998	0.0061998	0.0108978
Average	0.49918072	0.505361115	0.5022709	0.0022709	0.5064986	0.0064986	

b. $N_s = 8, f_w = 2$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49870636	0.50555571	0.502131	0.002131	0.5110072	0.0110072	0.0176853
2	0.49819933	0.50599439	0.5020969	0.0020969	0.5084628	0.0084628	0.0125514
3	0.49888227	0.50624811	0.5025652	0.0025652	0.5109883	0.0109883	0.0157251
4	0.49912602	0.50571385	0.5024199	0.0024199	0.5111247	0.0111247	0.0163524
5	0.4996572	0.5051419	0.5023996	0.0023996	0.5110628	0.0110628	0.0163108
6	0.49878538	0.50665322	0.5027193	0.0027193	0.5119823	0.0119823	0.0172604
7	0.49903478	0.50550432	0.5022696	0.0022696	0.5102407	0.0102407	0.0153966
8	0.4986346	0.50515685	0.5018957	0.0018957	0.5119636	0.0119636	0.0186969
9	0.49884751	0.50540094	0.5021242	0.0021242	0.5081833	0.0081833	0.0112803
10	0.4994794	0.50549011	0.5024848	0.0024848	0.5103879	0.0103879	0.0162172
11	0.49802428	0.50648732	0.5022558	0.0022558	0.508039	0.008039	0.0105507
12	0.49984907	0.50514029	0.5024947	0.0024947	0.5134565	0.0134565	0.0192871
Average	0.49893552	0.505707251	0.5023214	0.0023214	0.5105749	0.0105749	

(Cont.) c. $N_s = 8, f_w = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49944543	0.50501754	0.5022315	0.0022315	0.5131801	0.0131801	0.0165447
2	0.49923084	0.50618961	0.5027102	0.0027102	0.5163098	0.0163098	0.0195086
3	0.49948915	0.50595748	0.5027233	0.0027233	0.5164627	0.0164627	0.0188129
4	0.49791586	0.50627818	0.502097	0.002097	0.5146064	0.0146064	0.0174466
5	0.49954359	0.50502469	0.5022841	0.0022841	0.5135391	0.0135391	0.0154008
6	0.49913236	0.50500461	0.5020685	0.0020685	0.5119023	0.0119023	0.0156807
7	0.49901769	0.50609031	0.502554	0.002554	0.512231	0.012231	0.0138531
8	0.4997669	0.50593613	0.5028515	0.0028515	0.5137721	0.0137721	0.0169249
9	0.49918897	0.50589985	0.5025444	0.0025444	0.5140257	0.0140257	0.0168013
10	0.49748344	0.50518901	0.5013362	0.0013362	0.5116034	0.0116034	0.013842
11	0.49916516	0.50598235	0.5025738	0.0025738	0.513901	0.013901	0.01765
12	0.49942671	0.50573576	0.5025812	0.0025812	0.5170181	0.0170181	0.0203459
Average	0.49906718	0.505692127	0.5023796	0.0023796	0.514046	0.014046	

d. $N_s = 15, f_w = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49918766	0.50630671	0.5027472	0.0027472	0.5043665	0.0043665	0.0093878
2	0.49948799	0.50517088	0.5023294	0.0023294	0.5043665	0.0043665	0.0075608
3	0.499933	0.50504101	0.502487	0.002487	0.5041741	0.0041741	0.0066408
4	0.49973235	0.50567204	0.5027022	0.0027022	0.5048323	0.0048323	0.0083196
5	0.49961401	0.50525695	0.5024355	0.0024355	0.5045129	0.0045129	0.0085679
6	0.49974381	0.50503725	0.5023905	0.0023905	0.5034205	0.0034205	0.0053537
7	0.49937253	0.50517716	0.5022749	0.0022748	0.5050986	0.0050986	0.0109576
8	0.4988081	0.50502817	0.5019181	0.0019181	0.5041521	0.0041521	0.0068359
9	0.49881568	0.50502514	0.5019204	0.0019204	0.5046925	0.0046925	0.0119113
10	0.49947265	0.50502651	0.5022496	0.0022496	0.5040509	0.0040509	0.0074641
11	0.49930384	0.505237	0.5022704	0.0022704	0.5057298	0.0057298	0.011275
12	0.49934828	0.50505565	0.502202	0.002202	0.5041576	0.0041576	0.008513
Average	0.49940166	0.505252873	0.5023273	0.0023273	0.5044629	0.0044629	

e. $N_s = 15, f_w = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49900849	0.50560675	0.5023076	0.0023076	0.5091692	0.0091692	0.0237996
2	0.49949217	0.50514396	0.5023181	0.0023181	0.5098424	0.0098424	0.0159538
3	0.49939763	0.5050199	0.5022088	0.0022088	0.5077875	0.0077875	0.0129313
4	0.49886258	0.50511814	0.5019904	0.0019904	0.5080301	0.0080301	0.0137491
5	0.49901183	0.50516398	0.5020879	0.0020879	0.5087581	0.0087581	0.0135335
6	0.4992263	0.50507799	0.5021522	0.0021521	0.508966	0.008966	0.0145932
7	0.499079	0.5050282	0.5020536	0.0020536	0.5097317	0.0097317	0.0170256
8	0.49912469	0.50500875	0.5020667	0.0020667	0.5080259	0.0080259	0.0128308
9	0.49921088	0.50532749	0.5022692	0.0022692	0.5092227	0.0092227	0.0152136
10	0.49896978	0.50518432	0.5020771	0.0020771	0.5079268	0.0079268	0.013086
11	0.49934466	0.50507624	0.5022105	0.0022105	0.5090934	0.0090934	0.0155553
12	0.49958023	0.50506043	0.5023203	0.0023203	0.5098407	0.0098407	0.015221
Average	0.49919235	0.505151346	0.5021719	0.0021719	0.5088662	0.0088662	

(Cont.)

f. $N_s = 15, f_w = 5$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49917667	0.50511148	0.5021441	0.0021441	0.5158601	0.0158601	0.019877
2	0.49928281	0.50566513	0.502474	0.002474	0.5145221	0.0145221	0.0177471
3	0.49959884	0.50543568	0.5025173	0.0025173	0.5122182	0.0122182	0.015674
4	0.49934423	0.50626777	0.502806	0.002806	0.5134236	0.0134236	0.0170806
5	0.49918115	0.5055491	0.5023651	0.0023651	0.5108567	0.0108567	0.01309
6	0.4994214	0.505051	0.5022362	0.0022362	0.511652	0.011652	0.0143348
7	0.4995079	0.50506496	0.5022864	0.0022864	0.5105026	0.0105026	0.013805
8	0.49886887	0.50534219	0.5023555	0.0023555	0.5121111	0.0121111	0.0151241
9	0.49961189	0.50533472	0.5024733	0.0024733	0.5129126	0.0129126	0.0159488
10	0.4998642	0.50500773	0.502436	0.002436	0.5120777	0.0120777	0.0161682
11	0.49814443	0.505244	0.5016942	0.0016942	0.5101859	0.0101859	0.0134742
12	0.4994554	0.50613233	0.5027939	0.0027939	0.5133779	0.0133779	0.0175827
Average	0.49928815	0.505475508	0.5023818	0.0023818	0.512475	0.012475	

g. $N_s = 15, f_w = 7$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49921147	0.50538233	0.5022969	0.0022969	0.5134034	0.0134034	0.0132934
2	0.49953193	0.50501715	0.5022745	0.0022745	0.5175264	0.0175264	0.0181207
3	0.49906269	0.50608634	0.5025745	0.0025745	0.5140893	0.0140893	0.015053
4	0.49908672	0.50518782	0.5021373	0.0021373	0.5142892	0.0142892	0.0159702
5	0.49928674	0.50512343	0.5022051	0.0022051	0.5149995	0.0149995	0.0159002
6	0.49862235	0.50561659	0.5021195	0.0021195	0.515003	0.015003	0.0146906
7	0.49888998	0.50509682	0.5019934	0.0019934	0.5168192	0.0168192	0.0173196
8	0.49914178	0.50505716	0.5020995	0.0020995	0.5161403	0.0161403	0.0165769
9	0.49955719	0.50555008	0.5025536	0.0025536	0.5170762	0.0170762	0.0183007
10	0.49903783	0.50568651	0.5023622	0.0023622	0.5153519	0.0153519	0.0159533
11	0.49961057	0.50528602	0.5024483	0.0024483	0.5149022	0.0149022	0.0171499
12	0.49966269	0.50500601	0.5023344	0.0023344	0.5173278	0.0173278	0.0181771
Average	0.49922516	0.505341355	0.5022833	0.0022833	0.5155774	0.0155774	

h. $N_s = 21, f_w = 1$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49931695	0.50538657	0.5023518	0.0023518	0.5041177	0.0041177	0.0065764
2	0.49944147	0.50534463	0.5023931	0.0023931	0.5045873	0.0045873	0.0087526
3	0.49977857	0.5051036	0.5024411	0.0024411	0.5043139	0.0043139	0.0092816
4	0.49929358	0.5052798	0.5022867	0.0022867	0.5043502	0.0043502	0.0077644
5	0.49942866	0.50511409	0.5022714	0.0022714	0.5037087	0.0037087	0.0060528
6	0.49906262	0.50505645	0.5020595	0.0020595	0.5042384	0.0042384	0.0094674
7	0.499474	0.50517891	0.5023265	0.0023265	0.5046626	0.0046626	0.0087634
8	0.49933806	0.50515041	0.5022442	0.0022442	0.503702	0.003702	0.006492
9	0.49967052	0.50521446	0.5024425	0.0024425	0.5044678	0.0044678	0.0092793
10	0.4990245	0.50535402	0.5021893	0.0021893	0.5039407	0.0039407	0.0095746
11	0.49925803	0.50530834	0.5022832	0.0022832	0.5041243	0.0041243	0.005212
12	0.4994118	0.5050493	0.5022306	0.0022306	0.5042305	0.0042305	0.0074928
Average	0.4993749	0.505211715	0.5022933	0.0022933	0.5042037	0.0042037	

(Cont.)

i. $N_s = 21, f_w = 3$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49926859	0.505432	0.5023503	0.0023503	0.5077672	0.0077672	0.0143445
2	0.49946094	0.50507853	0.5022697	0.0022697	0.5061471	0.0061471	0.0095461
3	0.49945736	0.50508226	0.5022698	0.0022698	0.5077436	0.0077436	0.0143946
4	0.49877968	0.50508729	0.5019335	0.0019335	0.505906	0.005906	0.0108534
5	0.49949521	0.50516978	0.5023325	0.0023325	0.5074487	0.0074487	0.0139683
6	0.49875437	0.50524475	0.5019996	0.0019996	0.5057807	0.0057807	0.0094563
7	0.49942347	0.50507701	0.5022502	0.0022502	0.5063741	0.0063741	0.0105449
8	0.49917074	0.50544583	0.5023083	0.0023083	0.5064334	0.0064334	0.0113358
9	0.49930134	0.50543647	0.5023689	0.0023689	0.5078356	0.0078356	0.0136095
10	0.49986855	0.50509117	0.5024799	0.0024799	0.5070241	0.0070241	0.0122775
11	0.49910589	0.50510477	0.5021053	0.0021053	0.5067803	0.0067803	0.0123945
12	0.49924919	0.50507019	0.5021597	0.0021597	0.5085411	0.0085411	0.0155197
Average	0.49927794	0.505193338	0.5022356	0.0022356	0.5069818	0.0069818	

j. $N_s = 21, f_w = 5$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.4989793	0.50524254	0.5021109	0.0021109	0.5097801	0.0097801	0.013795
2	0.49941933	0.50533127	0.5023753	0.0023753	0.5106901	0.0106901	0.0149554
3	0.4990898	0.50609337	0.5025916	0.0025916	0.5094831	0.0094831	0.0134307
4	0.49944274	0.50516118	0.502302	0.002302	0.509493	0.009493	0.0132579
5	0.49923013	0.5051598	0.502195	0.002195	0.5083682	0.0083682	0.013119
6	0.49915662	0.50502213	0.5020894	0.0020894	0.5102857	0.0102857	0.0149801
7	0.49910243	0.50595729	0.5025299	0.0025299	0.510803	0.010803	0.0157481
8	0.49951605	0.50510798	0.502312	0.002312	0.5101004	0.0101004	0.0142097
9	0.49949946	0.5052413	0.5023704	0.0023704	0.5097258	0.0097258	0.0141988
10	0.49948672	0.50514715	0.5023169	0.0023169	0.509692	0.009692	0.01531
11	0.49923988	0.50548895	0.5023644	0.0023644	0.5108094	0.0108094	0.0162715
12	0.49943569	0.50532238	0.502379	0.002379	0.5104157	0.0104157	0.0151396
Average	0.49929985	0.505356278	0.5023281	0.0023281	0.5099705	0.0099705	

k. $N_s = 21, f_w = 7$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.49893459	0.50549386	0.5022142	0.0022142	0.5127394	0.0127394	0.0152902
2	0.49926369	0.50504884	0.5021563	0.0021563	0.512814	0.012814	0.0172981
3	0.49937003	0.50512409	0.5022471	0.0022471	0.5150132	0.0150132	0.0186415
4	0.49918837	0.50503421	0.5021113	0.0021113	0.513678	0.013678	0.0174939
5	0.49932718	0.50504101	0.5021841	0.0021841	0.5141152	0.0141152	0.0175016
6	0.49931434	0.50532103	0.5023177	0.0023177	0.5132846	0.0132846	0.0164223
7	0.4992437	0.50500753	0.5021256	0.0021256	0.512574	0.012574	0.0169267
8	0.49934622	0.50508527	0.5022158	0.0022157	0.5134157	0.0134157	0.0177348
9	0.49897656	0.50542936	0.502203	0.002203	0.5128614	0.0128614	0.0163791
10	0.49939036	0.50513913	0.5022648	0.0022647	0.5119821	0.0119821	0.0150841
11	0.49897199	0.50513231	0.5020522	0.0020522	0.5144925	0.0144925	0.0186373
12	0.49954425	0.50502574	0.502285	0.002285	0.5099386	0.0099386	0.0121739
Average	0.49923927	0.505156865	0.5021981	0.0021981	0.5130757	0.0130757	

(Cont.)

l. $N_s = 21, f_w = 10$

Treatment	FTSIA				Statistical Calculations		
	Low bound	High bound	Mean	Error	Mean	Error	Std.
1	0.4997133	0.50534933	0.5025313	0.0025313	0.5136752	0.0136752	0.0141343
2	0.4997523	0.50542793	0.5025901	0.0025901	0.5149228	0.0149228	0.0150769
3	0.49949133	0.50504049	0.5022659	0.0022659	0.5146835	0.0146835	0.0155115
4	0.49932866	0.50505773	0.5021932	0.0021932	0.5203112	0.0203112	0.0198055
5	0.49885099	0.50566457	0.5022578	0.0022578	0.5153462	0.0153462	0.0157767
6	0.49940298	0.50595539	0.5026792	0.0026792	0.5147693	0.0147693	0.0147205
7	0.49924746	0.50519481	0.5022211	0.0022211	0.5156894	0.0156894	0.0151724
8	0.49966613	0.50504135	0.5023537	0.0023537	0.5150388	0.0150388	0.0152035
9	0.49941179	0.5057349	0.5025734	0.0025734	0.515244	0.015244	0.0143989
10	0.49897576	0.50547452	0.5022251	0.0022251	0.5144501	0.0144501	0.015345
11	0.49882502	0.50539256	0.5021088	0.0021088	0.5162709	0.0162709	0.0177374
12	0.49930065	0.50501257	0.5021566	0.0021566	0.513945	0.013945	0.0146133
Average	0.49933053	0.505362179	0.5023464	0.0023464	0.5153622	0.0153622	

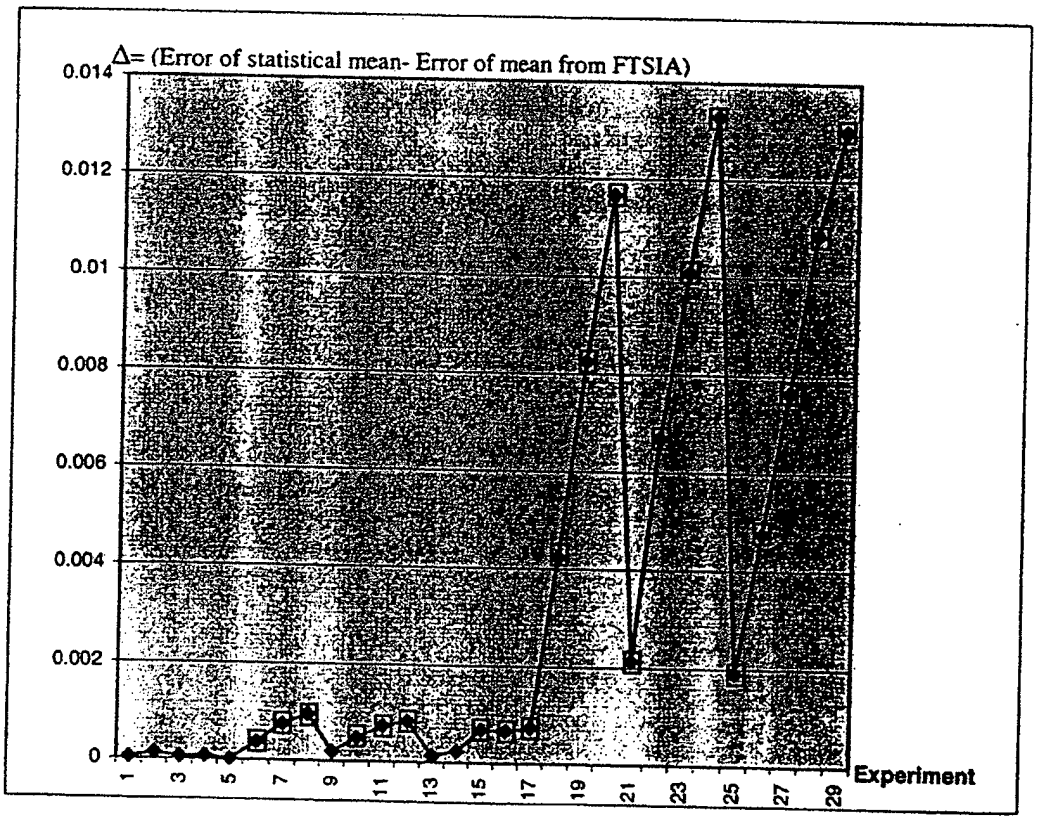


Figure C.1 Differences of Error of Statistical Mean and Error of Mean from FTSIA in 29 Experiments with 12 Treatments each (Experiments 1 to 29 correspond to Tables C.1.a to C.3.l respectively)

* the point surrounded by a rectangular means the difference is significant (α=95%)in the corresponding experiment

Distributed Micro Flow-Sensor Arrays and Networks: Design of Architectures and Communication Protocols

Y. Liu* and S.Y. Nof**
*School of Industrial Engineering, Purdue University, IN 47906
**Corresponding author, School of Industrial Engineering, Purdue University, IN 47906
Tel: (765)494-5427 Fax: (765)-494-1299 Email: nof@ecn.purdue.edu

Abstract: Distributed micro flow-sensor arrays and networks (DMFSA/N) are built from collections of spatially scattered, intelligent micro flow-sensor nodes. It can enhance the reliability and fault-tolerance of the system. A new cluster network architecture, and two alternatives of fault-tolerant time-out communication protocol (FTTP) are proposed in this paper. TIE (Teamwork Integration Evaluator)/MEMS, a MEMS sensor network simulation tool, is developed to evaluate the effectiveness of the network architectures and communication protocols.

Keywords: MEMS sensors, fault-tolerant integration, time-out, modeling, teamwork integration evaluator

1. Introduction

The measurement of fluid flow is an important technology used in nearly every industry. Flow-sensors are used to measure the movement of fluid flow (liquid or gas flow). They are widely used in a variety of applications, such as automotive applications, environmental monitoring, biomedical diagnostics, and industrial testing.

Before the advent of micro minimization, sensor systems tended to be bulky and expensive, so single sensor systems have typically been designed to save space and manufacturing cost. Although single sensor systems are relatively easy to construct and analyze, they have limitations. For instance, if we need to measure several variables, say,

temperature, pressure and flow density, all at the same time, single sensor systems cannot be used. In addition, a single sensor cannot guarantee to deliver accurate information all the time, because it is inevitably affected by noise, refresh delay, and other uncertain disruptions. Therefore, single sensor systems are not suitable for critical applications.

Development of MEMS has enabled production of large amounts of micro sensors at low cost. Microelectromechanical system (MEMS) technology deals with design and fabrication of entire electrical and mechanical systems, usually on a single silicon chip. Micro flow-sensors are one of the most common MEMS devices. Due to their small sizes, micro flow-sensors have been developed in recent years (Berberig et al., 1998; Okulan et al., 2000; Wu and Sansen, 2002) with the following advantages:

1. Interfere less with the environment they are measuring
2. Lower manufacturing cost
3. Can be applied in narrow spaces, such as inside living organisms, small pipes, and automobile engines.
4. Multiple sensors can be designed with redundancy to improve information accuracy, responsiveness, and fault tolerance of the system.

Distributed micro flow-sensor arrays and networks (DMFSA/N) are built from collections of spatially scattered, intelligent micro flow-sensor nodes. Each node has the ability to measure the local flow within its accuracy limits, process the raw sensor data, and cooperate with its neighboring nodes. It enables more reliable and timely monitoring and control. Sensors incorporated with dedicated signal processing functions are called intelligent sensors, or smart sensors. The main roles of dedicated signal processing are to enhance design flexibility and realize new sensing functions. Additional roles are to

reduce loads on central processing units and signal transmission lines by distributing information processing to the lower layers of the system (Yamasaki, 1995). Here, the micro flow-sensor array refers to a group of the same type of micro flow-sensors which are placed close together to measure the same variable of interest. Alternatively, sensors with different measurement functions can be combined in an array. Different groups of micro flow-sensor arrays can be distributed widely in the flow environment to form a micro flow-sensor network.

Developing an effective DMFSA/N, however, also faces great challenges:

1. First, since sensor nodes are distributed spatially, appropriate sensor network architecture has to be designed to support the efficient communication among the sensor nodes
2. Second, minimizing communication cost, C, which is often related to communication time, c_t, and energy consumption, c_e, is important in online control and energy constrained applications. Therefore, cost-efficient communication protocols need to be developed
3. In addition, it is unrealistic to expect all the micro sensor nodes and communication links to be fault-free and available all the time, therefore, developing fault-tolerant communication protocols is another important issue in the design of DMFSA/N.

The objective of this paper is to address the design of the network architectures, *NA*, and communication protocols, *CP*, for DMFSA/N. A new cluster network architecture, two alternatives of fault-tolerant time-out communication protocol (FTTP) – one with blind stations, the other with smart base stations, are proposed in this paper (Nof and Liu,

2003). TIE (Teamwork Integration Evaluator)/MEMS, a MEMS sensor network simulation tool, has been developed to evaluate the effectiveness of the network architectures and communication protocols. Experimental results show that the proposed cluster architecture can support efficient communication in the sensor network, and the FTTP with smart base stations requires less communication time than the FTTP with blind stations, although smart base stations have more complex structures than blind base stations. In this paper, some current research on the design of sensor network architectures and communication protocols is reviewed in section II; the new cluster architecture and FTTP for DMFSA/N are presented in section III; in section IV, TIE/MEMS is described. Two sets of simulation experiments were conducted using TIE/MEMS to evaluate the five combinations of sensor network architectures and communication protocols, and the two alternatives of FTTP. The experimental results and discoveries are presented and discussed. Finally, conclusions, significance of the discoveries, and future challenges are discussed in section V.

2. Sensor Network Architecture and Communication Protocol Review

The abstraction of a sensor network can be represented as $NA = (V, E)$, where V is the set of sensor node and E is the set of communication links connecting the sensor nodes. $V = \{v_i\}(i = 1,2,..., N_s)$, where N_s is the number of sensor nodes in the network and v_i is the sensor node i. $E = \{e_{ij}\}(i = 1,2,..., N_s; j = 1,2,..., N_s)$ where e_{ij} is the link between v_i and v_j.

The two most conventional network architectures that can be used for sensor networks are the *committee architecture* (COA), and the *hierarchical architecture* (HIA), first discussed by Wesson *et al.* (1981). In COA, each node in the network is autonomous and can send information to all other nodes in the network. Thus if N_s is equal to n, it needs interconnections that are of the order $O(n^2)$. With such a high number of interconnections, this architecture is very robust to link failures because the messages from a source node to a destination node can be easily rerouted by bypassing the failed links. However, COA is not suitable for the sensor network with a large value of N_s, because it is too expensive with respect to the communication cost, C, and not easily extensible (figure 1(a)). In HIA, nodes are connected to form strict hierarchies. Each node (except the lowest-level leaf node) receives information from the lower level node; integrates the information received according to its position in the hierarchy, and then sends information to the node at the upper level. The root node is the node at the highest level and responsible for interpretation of data at the global level. A special case of HIA is a complete binary tree (figure 1(b)). Compared with COA, HIA is extensible more easily, but it is not tolerant to link failures. Two other sensor network architectures, proposed by Iyengar *et al.* (1990) and Iyengar *et al.* (1994) are *flat Tree architecture* (FTA) and *Multilevel binary de Bruijn architecture* (MBDA). In FTA, the network is partitioned into clusters and each cluster is internally organized as a binary tree. All the root nodes in the network are fully interconntected (figure 1(c)). FTA is not sufficiently robust to link failures. MBDA is a *multilevel network* (MLN) with the top level fully connected and with each of the other levels interconnected as a *de Bruijn network* (figure 1(d)). MBDA is not suitable for the DMFSA/N because of its complex structure.

In recent years, there has been increasing interest in the development of communication protocols for the distributed sensor network, especially the wireless sensor network (Gossink et al., 1998; Agre et al., 1999; Chandrakasan et al., 1999; Heinzelman et al., 1999; Bhushan and Rengaswamy, 2000; Min et al., 2002). The sensor network can improve the accuracy of information obtained via collaboration among sensor nodes.

The simplest communication protocol is the *point-to-point protocol* (PTP). In this protocol, sensor nodes A and B communicate exclusively with each other without interfering with other nodes (figure 2(a)). Another commonly used communication protocol is *broadcasting protocol* (BP). In this protocol, the information sent out by the sender node can be received by all its neighbors using only one transmission (figure 2(b)). Since a node always sends data to its neighbors, regardless of whether or not they have already received the data from another source, BP can lead to the implosion problem and waste resources by sending requested copies of data to the same node. In the *Gossiping protocol* (GP), instead of indiscriminately sending information to all its neighboring nodes, each sensor node only forwards the data to one randomly selected neighbor (Heinzelman, 1999) (see figure 2(c)). While GP requires less communication than BP, it is not as robust to link failures as BP because a node can only rely on one other node to reroute information for it if its link fails.

[Insert figure 1 here]

[Insert figure 2 here]

3. DMFSA/N Architecture and Communication Protocol

In this section, the proposed cluster network architecture and fault-tolerant time-out communication protocol for DMFSA/N are presented.

3.1 Cluster Network Architecture for DMFSA/N

The new cluster network architecture for DMFSA/N, *Cluster architecture* (CLA) is shown in figure 3. It is preferred because it overcomes the major limitations of both COA and HIA by limiting the amount of communication, while at the same time tolerating failures of the sensor nodes and links. In this structure, DMFSA/N are divided into *sensor cluster units* (SCUs), each of which consists of a set of intelligent micro flow-sensor nodes and a base station. The intelligent sensor nodes within the same SCU are also called *sibling nodes*. An intelligent micro flow-sensor node consists of a dedicated processing unit and an associated micro flow-sensor that measures the variables of interest. A base station is needed for each cluster to integrate information from the individual sensor nodes in the cluster. Considering the complexity of computation and energy constraints of the micro sensor node, a more powerful processor is selected for the base station. The base station acts as the control node of the respective SCU, controlling the signal transmission of the sensor nodes, managing information rerouting in case of link failures, and integrating the information from all the active sensor nodes in the SCU. To enable the system to be tolerant to link failures, a sensor node sends information not only to the base station, but also to one or more other sibling nodes that are also called its *backup nodes*. Two communication protocols can function among the sibling nodes: (1) BP, in which each sensor node broadcasts its information to all its sibling nodes; (2) *Evenly wide gossiping protocol* (EWGP), a revised GP (Liu and Nof, 2001a), in which

instead of sending information to only one other sibling node by random selection according to GP, the backup nodes are chosen in the way that the roles of backup nodes are evenly distributed among the sibling nodes, so it will not happen that some nodes assume too many 'duties', while others are 'starved'.

The two communication protocols, GP and EWGP, can also be used for the communications among the base stations. The external processor is the commander of the entire network, in which information from all the SCUs is integrated for decision making as required. The clusters can be deployed along the flow path, depending on the flow situation and applications.

[Insert figure 3 here]

3.2 Fault-tolerant Time-out Communication Protocol

The design of fault-tolerant time-out protocol (FTTP) is motivated by the combination of time-out and task coordination protocols (Esfarjani and Nof, 1998; Williams et al., 2002, Peralta et al., 2003), and the need for fault-tolerant integration (Liu and Nof, 2001b). At the beginning of each data transmission cycle, the base station broadcasts the data transmission request to all the sensor nodes in the SCU. Receiving the request, each sensor node sends its data to the buffer inside the base station. Each sensor node has a corresponding buffer in the base station to store its data so that the base station can trace the source of the data. Then the data will be retrieved into the processing unit in the base station. According to FTTP, the base station will stop waiting for the information from the sensor node if a certain amount of time, T, has passed, where T is the pre-determined time-out period and its value, which can also be adaptable, depends on the application.

The base station will then announce the rerouting task message to the sibling nodes of the node with failed link. Two alternatives exist for where the base station should send the task announcement: (1) with a Blind Base Station, the message is sent to all the sibling nodes; (2) with a Smart Base Station, the message is only sent to the sibling node that can finish the task earliest, based on the current status of the nodes in the system.

3.2.1 FTTP with Blind Base Station

In this protocol, when the link of a sensor node, say node v_k, fails, the base station will broadcast the rerouting task message (RTM) for v_k to all its sibling nodes. Receiving RTM, its backup node b_{ki} (the ith backup node of v_k) generates a bid value bv_{ki} and sends it to the base station. $b_{ki} \in BK_k$, where BK_k is the set of backup nodes of v_k. bv_{ki} can be calculated as follows:

$$bv_{ki} = (m_{ki} + 1)\gamma_{ki} \tag{1}$$

where γ_{ki} is the transmission rate of node b_{ki} (time spent in transmitting one message) and m_{ki} is the number of tasks that have been offered to node b_{ki}. Because of the computation limit of the intelligent sensor node, it is unrealistic for it to keep track of its transmission rate over time, γ_{ki} is assumed to be constant for node b_{ki}. The base station then evaluates the priorities of the backup nodes based on their bid values (the lower the bid value, the higher the priority), and sends the task offer message (TOM) to the backup node with the highest priority, $b_{k(1)}$. Receiving the task offer message, $b_{k(1)}$ sends the required backup data message (BDM) to the base station.

The structures of the base station and intelligent nodes, and the logic chart of the FTTP with blind base station are shown in figure 4 and figure 5 respectively.

[Insert figure 4 here]

[Insert figure 5 here]

3.2.2 FTTP with Smart Base Station

In this protocol, in order to limit the unnecessary communication and reduce the communication traffic, instead of broadcasting RTM for v_k, whose link fails, to all its sibling nodes, the base station first evaluates the priorities of the its backup nodes and then sends TOM only to the node that can finish the task the earliest. The time when the backup node b_{ki} finishes the task, t_{ki}, can be calculated as, assuming the transmission starts at time 0:

$$t_{ki} = (m_{ki} + 1)\gamma_{ki} \tag{2}$$

where m_{ki} is the number of tasks that have been offered to the backup node b_{ki}, and γ_{ki} is the transmission rate of b_{ki} (time spent in transmitting one message). The base station then sends TOM to the node $b_{k(1)}$ whose $t_{k(1)}$ is the smallest. In order to find out t_{ki}, the base station needs to have a knowledge base which keeps track of the information for each node – its backup nodes, transmission rate, and the number of tasks that have been offered. Figure 6 shows the structures of base station and intelligent sensor nodes for the FTTP with smart base station. The structure of the knowledge base is shown in figure 7, in which n is the number of sensor nodes in the cluster and the backup nodes are

hypothetical for illustration only. Figure 8 shows the logic chart of the FTTP with smart base station.

Compared with the FTTP with blind base station, the FTTP with smart base station requires less communication, and has the advantage of being able to keep track of the updated transmission rate of each sensor node, but it requires a base station with more complex structure (with an additional knowledge base). Therefore, a tradeoff has to be considered in order to determine when to apply each protocol.

[Insert figure 6 here]

[Insert figure 7 here]

[Insert figure 8 here]

4. TIE (Teamwork Integration Evaluator)/MEMS

TIE (Teamwork Integration Evaluator)/MEMS is developed in this research as a new version of the TIE simulators developed by the PRISM group (Khanna and Nof, 1994; Huang and Nof, 1996; Anussornnitisam and Nof, 2000; Anussornnitisam, 2003; Liu and Nof, 2001c) at the School of Industrial Engineering in Purdue University. The objective of TIE/MEMS is to simulate and evaluate the architectures and communication protocols developed for the distributed micro sensor network. TIE/MEMS is programmed with MPI (Message Passing Interface). MPI is not a new programming language; rather it is a library of subprograms that can be called from C and Fortran 77 programs. Since its completion in June 1994, MPI has received widespread acceptance because it is based on message passing, one of the powerful and widely used paradigms of programming parallel systems (Snir et al., 1996). MPI can be used to simulate different communication

schemes among the sensor nodes, such as point-to-point communication, collective communication, group communication, and so on (Liu and Nof, 2001c, Liu and Nof, 2001d; Huang and Nof, 2002).

4.1 Structure of TIE/MEMS

The overall structure of TIE/MEMS is depicted in figure 9. The inputs of TIE/MEMS are:

(1) micro sensor network architecture, *NA*;

(2) communication protocol, *CP*;

(3) sensor network parameters (such as the number of micro sensor nodes, N_s, the number of sensor nodes in each cluster, N_c, the number of failed links, N_l, etc.);

(4) constraints unique to the given MEMS sensors, e.g., physical and chemical constraints causing micro-stress, proximity noise, and other disturbances to measurement and communication.

TIE/MEMS consists of a collection of libraries, each of which is built to simulate one type of sensor network architecture and communication protocol. After the required inputs are entered, the corresponding library will be called to execute the program. The output of TIE/MEMS is communication cost, *C*, which is represented by the communication time c_t, or/and energy consumption c_e. If *C* is too high and can be reduced, *NA* or *CP*, or both need to be revised. TIE/MEMS can be extended by adding libraries for the newly developed *NA* and *CP*. Table 1 lists some examples of constraints related to MEMS sensors.

[Insert figure 9 here]

[Insert table 1 here]

4.2 Simulation Experiments and Results

Two sets of experiments were conducted to evaluate the sensor network architectures and communication protocols described above using TIE/MEMS. The experiments were implemented on the SGI Origin 2000 workstation in the Department of Computer Science at Purdue University.

4.2.1 Experiment with Network Architectures and Protocols

In this experiment set, five combinations of sensor network architectures and communication protocols were evaluated. They are HIA & PTP, COA & PTP, COA & BP, CLA & BP and CLA & EWGP with the number of backup nodes, N_b, equal to one. A total of thirty experiments were run for each of the five combinations and the outputs of the experiments are c_t and c_e per communication round. The result of c_t (sample mean) vs. N_s is shown in table 2. Table 3 shows the results of c_e vs. N_s (assuming the transmitting cost is 600mv per message transmission and the receiving cost is 200mw per message reception).

Statistical analysis results (α=95%) show that when N_s is small (N_s =3), there is no significant difference of c_t between HIA & PTP and CLA & EWGP, and they have the least c_t. There is no significant difference between COA & BP and CLA & BP, and COA & PTP has the greatest c_t among the five combinations. When N_s increases, however, c_t of the COA network, especially, the COA & PTP, increases dramatically, and CLA & EWGP has the least c_t. HIA networks have the least c_e, followed by CLA networks, and COA networks have the highest c_e. Therefore, COA and CLA are preferred in relatively small sensor networks because of their fault-tolerant characteristics, and HIA and CLA

are favored in large sensor networks due to their cost efficiency. CLA is a good choice in either case.

[Insert table 2 here]
[Insert table 3 here]

4.2.2 Experiments with FTTP

In this experiment set, FTTP with blind base station and FTTP with smart base station were evaluated. N_c and N_l are inputs of the experiment. Thirty experiments were run and the output is c_t per communication round. Table 4 summarizes the experiment results, which show, as expected, that the FTTP with blind base station requires significantly (α=95%) less evaluating time, c_{et}, than the FTTP with smart base station. The reason for this is that in the FTTP with blind base station, the bid values are calculated by the intelligent sensor nodes locally before they are sent to the base station, and the base station only evaluates the bid values and chooses the sensor node with the minimal bid value. Much more information transmission between the sensor nodes and the base station, however, is involved in the FTTP with blind base station. Because the information transmission time, c_{tt}, is much longer than c_{et} that requires only computation, the total communication time, c_t, which is the sum of c_{et} and c_{tt}, of the FTTP with blind base station is significantly longer than that of the FTTP with smart base station. The difference of c_t becomes more evident when N_c and N_l increase. On the other hand, the structure of the smart base station is more complex than that of the blind base station, as shown in figure 4 and figure 6. Figure 10 shows the ratio of c_t of the FTTP with blind base station to the FTTP with smart base station, r_{bs}, vs. N_c when N_l equal to 1, 2 and 3.

[Insert table 4 here]

[Insert figure 10 here]

5. Conclusions and Contributions of the Research

MEMS technology is the merger of the integrated circuit (IC) world with the mechanical world. It allows the integration of the sensors and actuators with IC at an unprecedented level. MEMS flow-sensor can be used in the applications in which conventional sensor systems cannot or are not suitable. DMFSA/N, built from a collection of cooperating intelligent micro flow-sensor nodes, can improve the reliability and fault-tolerance of the system.

5.1 Array/Network Architecture Design Observation

The appropriate sensor architecture for DMFSA/N depends on the number of sensor nodes needed in the system to provide accurate and timely information. When the number of sensor nodes in the system is relatively small, COA and CLA are better architectures because of their fault-tolerant characteristics. Otherwise, HIA and CLA are preferred in order to limit the communication cost. For example, table 2 shows that when N_s is greater than 15, C_t of COA & BP is more than 6 times of C_t of HIA & PTP, and C_t of COA & PTP is more than 30 times of C_t of HIA and PTP. The proposed CLA can support efficient information communication and integration in the system, because it not only requires less communication cost, but also tolerates a certain level of sensor and link failures. For example, table 2 shows that when N_s is greater than 15, C_t of CLA & BP is less than 1/4 of C_t of HIA & PTP, and C_t of CLA and EWGP (with N_b equal to 1) is less

than 1/8 of C_t of HIA & PTP. In addition, if the flow environment is not steady or non-homogeneous along the path, which is very common in the real flow measurement, CLA is better than the other alternatives because the clusters can be easily deployed along the path.

5.2 Protocol Design Observation

The best communication protocols depend on the number of sensor nodes in the system, sensor constraints and requirement of response time of the system. Although BP has better fault tolerance than EWGP, it requires relatively more communication. For example, Table 3 shows that when N_s is equal to15 (N_c is equal to 5), C_t of CLA and BP is almost twice C_t of CLA and EWGP, and the difference increases when N_s increases. Therefore, BP is suitable for the system in which the sensors or other IC components are not very robust because it has better fault-tolerance. Compared with the FTTP with blind base station, the FTTP with smart base station requires less communication time, but it demands the base station to have an additional knowledge base that keeps track of the information of the sensor nodes. Therefore, FTTP with smart base station is preferred in the on-line control applications, in which response time is critical.

In order to evaluate the effectiveness of micro sensor network architectures and communication protocols, TIE/MEMS has been developed using MPI in this research. TIE/MEMS can be expanded by adding libraries for the new developed sensor network architectures and communication protocols.

This research has been undertaken in response to actual problems encountered in the flow measurement area. Given the widespread use and importance of flow-sensors, the limitations of the current flow measurement technology are surprising. DMFSA/N opens original new directions for this area. Some of the main contributions of this research are:

1. Intelligent sensors manufactured using advanced IC technology will fundamentally change the nature of sensing and control systems, and modeling computational tool, such as TIE/MEMS, to evaluate their performance are useful.
2. The ability to place and activate the micro flow-sensors where and when they are needed enables better control of the flow process.
3. Redundancy enables more reliable monitoring and control because occurrence of failures of some sensors will not inhibit the sensing ability of the system.

For contributions 2 and 3, a FTTP is necessary to enable timely integration of the distributed sensors' readings. For future research, TIE/MEMS is being further developed. MEMS flow-sensor physical constraints (as shown in Table 1) are being quantified and incorporated into the simulation evaluation of the sensor network architectures and communication protocols.

Acknowledgement

This research is supported in part by Indiana 21st Century Research and Technology Fund, and by ATC Inc., a flow-sensor manufacturer.

References:

1. Agre, J.R., Clare, L.P., Pottie, G.J. and Romanov, N.P., 1999, "Development platform for self-organizing wireless sensor networks", Proceedings of SPIE - the International Society for Optical Engineering, Bellingham, WA, vol. 3713, pp.257-268.
2. Anussornnitisam, P., 2003, Design of Active Middleware Protocols for Coordination of Distributed Resources, Ph.D. Dissertation, School of IE, Purdue University, W. Lafayette, IN 47907.
3. Anussornnitisam, P., and Nof, S.Y., 2000, "a Teamwork Integration Evaluator for coordination protocols", *Proceedings of ICPR-2000*, Bangkok, Thailand.
4. Berberig, O., Nottmeyer, K., Mizuno, J., Kanai, Y., and Kobayashi, T., 1998, "Prandtl micro flow sensor (PMFS): A novel silicon diaphragm capacitive sensor for flow-velocity measurement", *Sensors & Actuators A-Physical.* 66(1-3) pp. 93-98.
5. Bhushan, M., and Rengaswamy, R., 2000, "Design of sensor location based on various fault diagnostic observability and reliability criteria", Computers & Chemical Engineering, 24(2), 2000, pp.735-741.
6. Chandrakasan, A., Amirtharajah, R., Seonghwan, Cho, Goodman, J., Konduri, G., Kulik, J., Rabiner, W. and Wang, A., 1999, "Design considerations for distributed microsensor systems", Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, Piscataway, NJ, pp.279-286.
7. Esfarjani, K., and Nof, S.Y., 1998, "A client-server model for integration and collaboration in production testing", International Journal of production Research, 36 (2), pp. 3925-3321.
8. Gossink, D.E., Scholz, J.B., and Gill, M.C., 1998, "Communication architecture to support distributed sensors", Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers, Piscataway, NJ, Part 1, vol.1, pp.588-592.
9. Heinzelman, W.R., Kulik, J., and Balakrishnan, H., 1999, "Adaptive protocols for information dissemination in wireless sensor networks", MobiCom'99,

Proceedings of Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM. New York, NY, pp.174-185.

10. Huang, C.Y., and Nof, S.Y., 1996, "TIE: Teamwork Integration Evaluation Simulator: A manual for TIE: 1.2", *Research Memorandum*, No 96-2, School of Industrial Engineering, Purdue University, West Lafayette, IN.
11. Huang, C.Y., and Nof, S.Y., 2002, "Evaluation of Agent-based Manufacturing Systems Based on a Parallel Simulator", Computers and IE, 43, pp. 529-552.
12. Iyengar, S.S., Jayasimha, D.N., and Nadig, D., 1994, "A versatile architecture for the distributed sensor integration problem", IEEE Transactions on Computers, 43(2), pp.175-185.
13. Iyengar, S.S., Kashyap, R.L., Madan, R.N., and Thomas, D., 1990, "A tree architecture for distributed sensor fusion problems", Proceedings of SPIE, Technical Symposium on Sensor Fusion, Orlando, FL, pp.126-135.
14. Khanna, N., and Nof, S.Y., 1994, "TIE: Teamwork Integration Evaluation Simulator: A preliminary User Manual for TIE 1.1", *Research Memorandum*, No. 94-21, School of Industrial Engineering, Purdue University, West Lafayette, IN.
15. Liu, Y., 2001b, "Distributed micro flow sensor arrays and networks: design of architecture and fault-tolerant integration", MSIE Thesis, School of Industrial Engineering, Purdue University, West Lafayette, IN.
16. Liu, Y., and Nof, S.Y., 2001a, "Distributed micro flow-sensor network design and modeling", Proceedings of IFAC Workshop on Manufacturing, Modeling for Management and Control, Prague, C.R., pp. 161-166.
17. Liu, Y., and Nof, S.Y., 2001c, "TIE/MEMS: modeling and analysis of micro sensor networks, user manual", Research Memorandum, Industrial Engineering, Purdue University, 01-01.
18. Liu, Y., and Nof, S.Y., 2001d, "Distributed micro flow-sensor network design and modeling", PRSIM Symposium, West Lafayette, IN.
19. Min, R., Bhardwaj, M., Cho, S-H. Ickes, N., Shih, E., Sinha, A., Wang, A., and Chandrakasan, A., 2002, "Energy-centric enabling technologies for wireless sensor networks", *IEEE Personal Communications.* 9(4), pp. 28-39

20. Nof, S.Y., and Liu, Y., 2003, "Fault-tolerant Time-out Communication Protocol and Sensor Appratus for Using same", Patent Pending, Purdue Ref: P-02028.P1. US
21. Okulan, N., Henderson, H., Thurman., A., and Chong, H., 2000, "Pulsed mode micromachined flow sensor with temperature drift compensation", *IEEE Transactions on Electron Devices.* 47(2), pp. 340-347.
22. Peralta, J., Anussornnitisarn, P., and Nof, S.Y., 2003, "Analysis of a Time-Out Protocol and Its Application in a Single Server Environment," International Journal of Computer Integrated Manufacturing, 16(1), 1-13.
23. Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W., and Dongarra, J., 1996, "MPI: The complete reference", MIT press.
24. Wesson, R., Hayes-Roth, F., Burge, W.J., Stasz, C., and Sunshine, C.A., 1981, "Network Structures for Distributed Situation Assessment", IEEE Transaction on System, Man and Cybernetics, 11(1), pp. 5-23.
25. Williams, N.P., Liu, Y., and Nof, S.Y., 2002, "TestLAN approach and protocols for the integration of distributed assembly and test networks", International Journal of Production Research, 40(17), pp. 4505-4522.
26. Wu, J. and Sansen, W., 2002, "Electrochemical time of flight flow sensor", *Sensors and Actuators, A: Physical.* 97-98, pp. 68-74.
27. Yamasaki, H., 1995, Handbook of Sensors and Actuators 3, Intelligent Sensors, Yokogawa Research Institute Corporation, Tokyo, Japan.

Tables:

Table 1. Examples of Constraints Related to MEMS Sensors

Factors	Examples
Scaling issues	Viscosity force; Surface effect
Problems in micromachining	Squeeze film damping; Particular contamination; Static stiction
Response	Decrease of response time due to protective gels, coatings or diaphragms
Reliability of microelectronic device	Breakdown of a wire-bond to a junction; Defective encapsulation; Defective semiconductor material; Thermal runaway
Life time	Aging; Contamination of the device; Limited power supply
Noise/interferences	Degraded signal due to the variance of temperature, humidity, etc.

Table 2. Communication Time (ms) of Five Combinations of Network Architectures, *NA*, and Communication Protocols, *CP*, vs. No. of Sensor Nodes, N_s

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	0.109	0.441	0.782	0.902	1.674
COA	PTP	0.725	6.153	22.920	40.894	93.200
COA	BP	0.471	2.233	4.835	9.217	20.555
CLA	BP	0.438[1]	0.161[2]	0.192[3]	0.205[4]	0.242[5]
CLA	EWGP (N_b=1)	0.112[1]	0.118[2]	0.115[3]	0.115[4]	0.121[5]

[1] N_c=3; [2] N_c=4; [3] N_c=5; [4] N_c=5; [5] N_c=6

Table 3. Energy Consumption (mw) of Five Combinations of Network Architectures, *NA*, and Communication Protocols, *CP*, vs. No. of Sensor Nodes, N_s

NA	*CP*	N_s				
		3	8	15	20	30
HIA	PTP	600*	600*	600*	600*	600*
		N/A	800**	1200**	1400**	1400**
		400***	800***	800***	800***	1600***
COA	PTP	4800	44800	168000	304000	696000
COA	BP	2400	15400	50400	87400	191400
CLA	BP	1000[1]	1200[2]	1400[3]	1400[4]	1600[5]
CLA	EWGP (N_b=1)	800[1]	800[2]	800[3]	800[4]	800[5]

(* c_e for the leaf nodes ; ** c_e for the intermediate nodes ; *** c_e for the root node; no intermediate nodes for HIA with when N_s is equal to 3; [1] N_c=3; [2] N_c=4; [3] N_c=5; [4] N_c=5; [5] N_c=6)

Table 4. Communication Time (μs), c_t, of the FTTP with Blind Base Station and FTTP with Smart Base Station vs. No. of Sensor Nodes in the Cluster, N_c, and No. of Failed Links, N_l

FTTP	N_l	N_c	c_{et}	c_{ct}	c_t
FTTP with Blind Base Station	1	4	2.17	142.77	144.93
		8	2.70	261.13	263.83
		15	2.90	477.37	480.27
		21	3.93	773.07	777.00
	2	4	3.13	346.60	349.73
		8	4.20	581.90	586.10
		15	5.87	1146.50	1152.37
		21	7.40	1706.13	1713.53
	3	4	4.63	556.80	561.43
		8	5.65	827.13	832.77
		15	7.70	1721.53	1729.23
		21	10.45	2587.52	2597.97
FTTP with Smart Base Station	1	4	4.33	22.43	26.77
		8	5.64	28.91	34.55
		15	7.16	33.06	40.23
		21	8.17	40.49	48.66
	2	4	7.90	45.33	53.23
		8	9.89	54.48	64.37
		15	13.90	58.50	72.40
		21	18.46	60.68	79.14
	3	4	10.60	62.13	72.73
		8	13.30	66.96	80.26
		15	16.83	70.69	87.51
		21	23.86	72.91	96.77

Figures:

Figure 1. Sensor Network Architectures: (a) Committee Architecture (b) Hierarchical Architecture (c) Flat Tree Architecture (d) Multilevel Binary de Bruijn Architecture

Figure 2. Communication Protocols: (a) Point-to-Point Protocol (b) Broadcasting Protocol (c) Gossiping Protocol

Figure 3.New Cluster Architecture for DMFSA/DMFSN (dash line means the link may or may not exist depending on the communication protocol used)

Figure 4. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Blind Base Station (dash line means the link may or may not exist depending on the protocol used.)

Figure 5. Logic Chart of the FTTP with Blind Base Station

Figure 6. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Smart Base Station (dash line means the link may or may not exist depending on the protocol used.)

Figure 7. Logic Chart of the FTTP with Smart Base Station

Figure 8. Logic Chart of the FTTP with Smart Base Station

Figure 9. Structure of TIE/MEMS (dash line- under development)

Figure 10. Ratio of Communication Time of the FTTP with Blind Base Station to the FTTP with Smart Base Station, r_{bs}, vs. No. of Sensor Nodes in the Cluster, N_c, when the No. of Failed Links, N_l, Equal to 1, 2 and 3

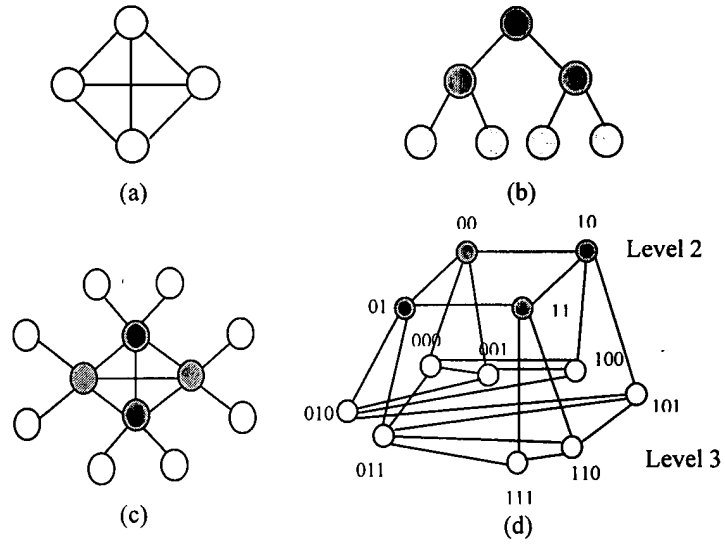


Figure 1. Sensor Network Architectures: (a) Committee Architecture (b) Hierarchical Architecture (c) Flat Tree Architecture (d) Multilevel Binary de Bruijn Architecture

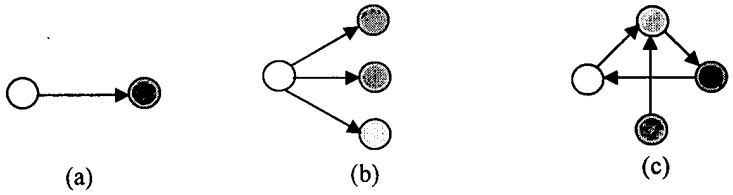


Figure 2. Communication Protocols: (a) Point-to-Point Protocol (b) Broadcasting Protocol (c) Gossiping Protocol

162

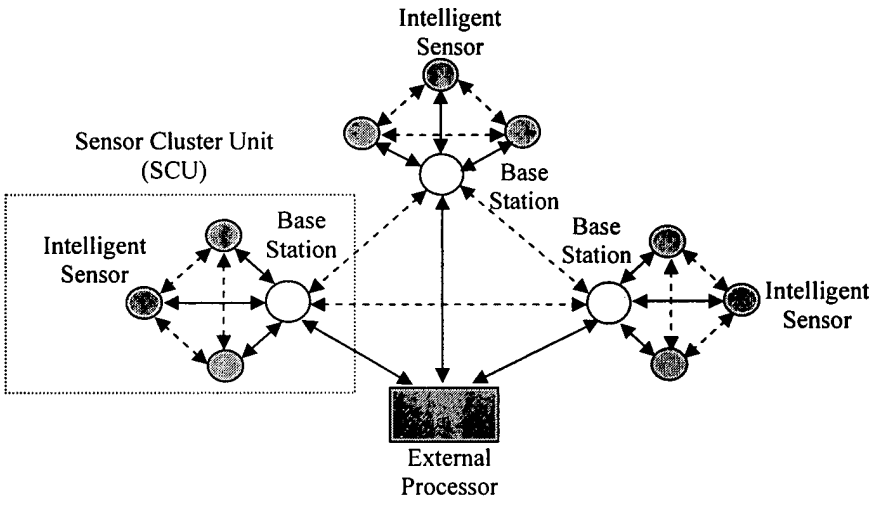


Figure 3.New Cluster Architecture for DMFSA/DMFSN (dash line means the link may or may not exist depending on the communication protocol used)

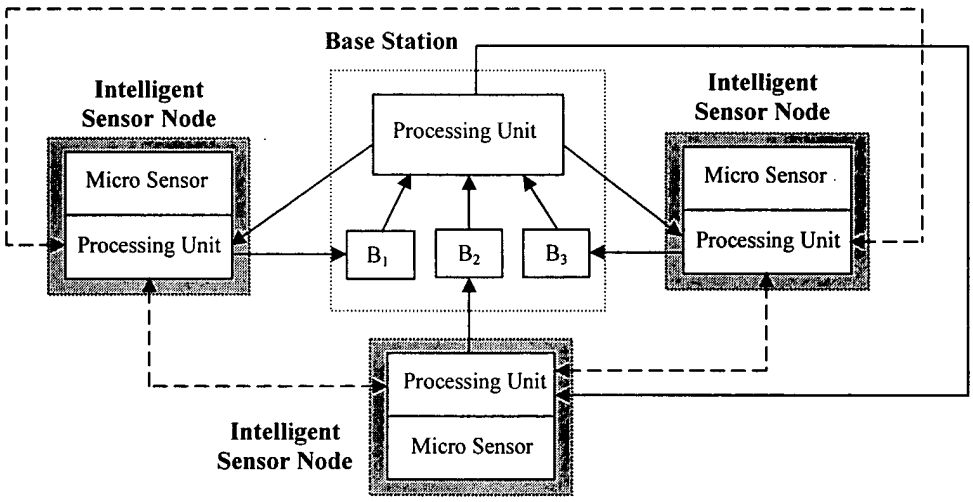


Figure 4. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Blind Base Station (dash line means the link may or may not exist depending on the protocol used.)

163

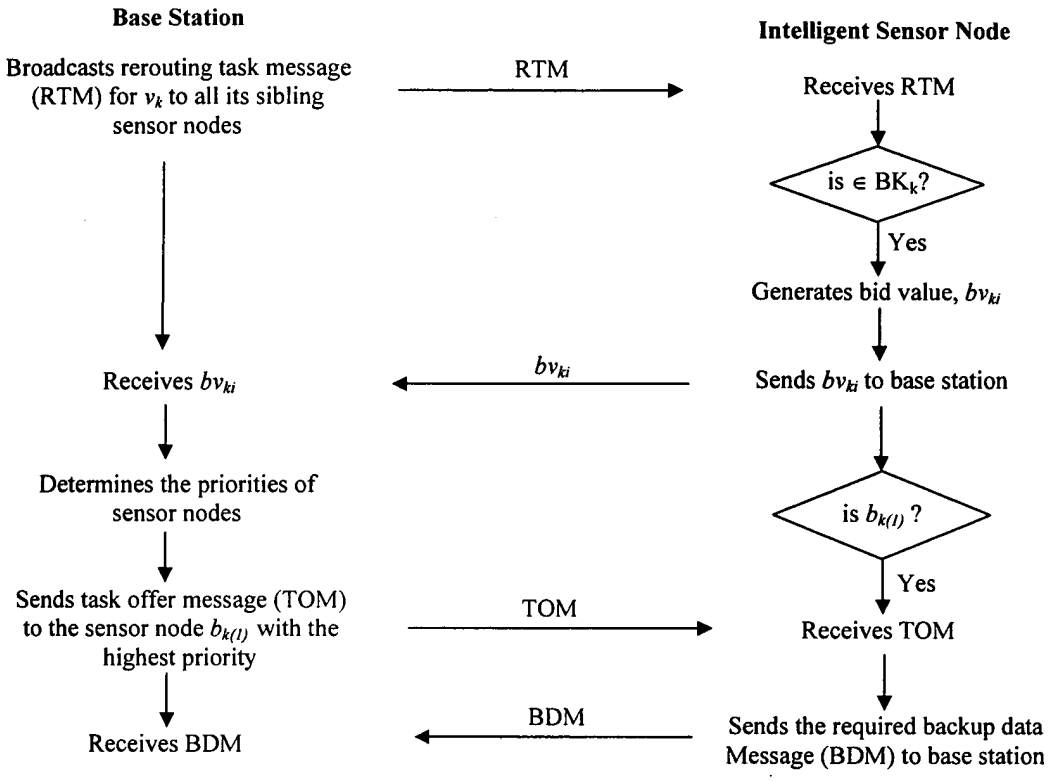


Figure 5. Logic Chart of the FTTP with Blind Base Station

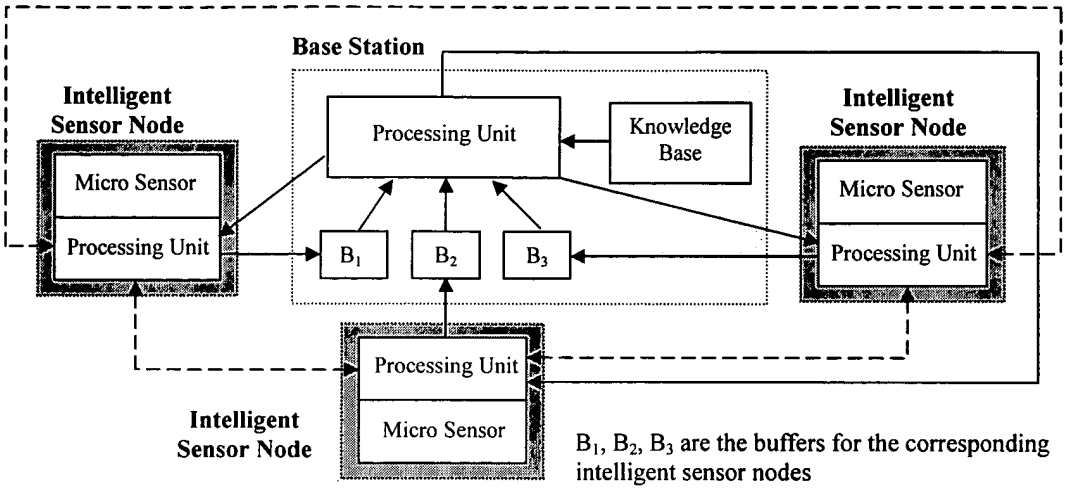


Figure 6. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Smart Base Station (dash line means the link may or may not exist depending on the protocol used.)

V	BK	γ	m
v_1	2, 3, 4	γ_1	m_1
v_2	1, 4, 5	γ_2	m_2
...	...	...	...
v_n	3, n-2, n-1	γ_n	m_n

Knowledge Base

Figure 7. Logic Chart of the FTTP with Smart Base Station

165

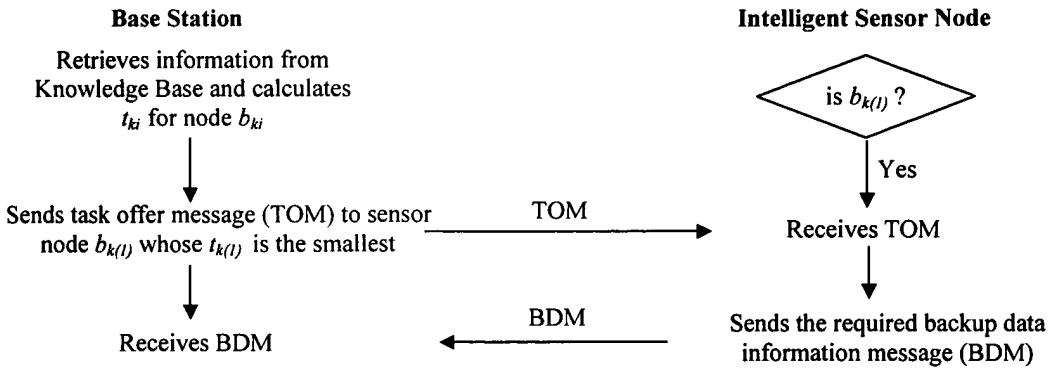


Figure 8. Logic Chart of the FTTP with Smart Base Station

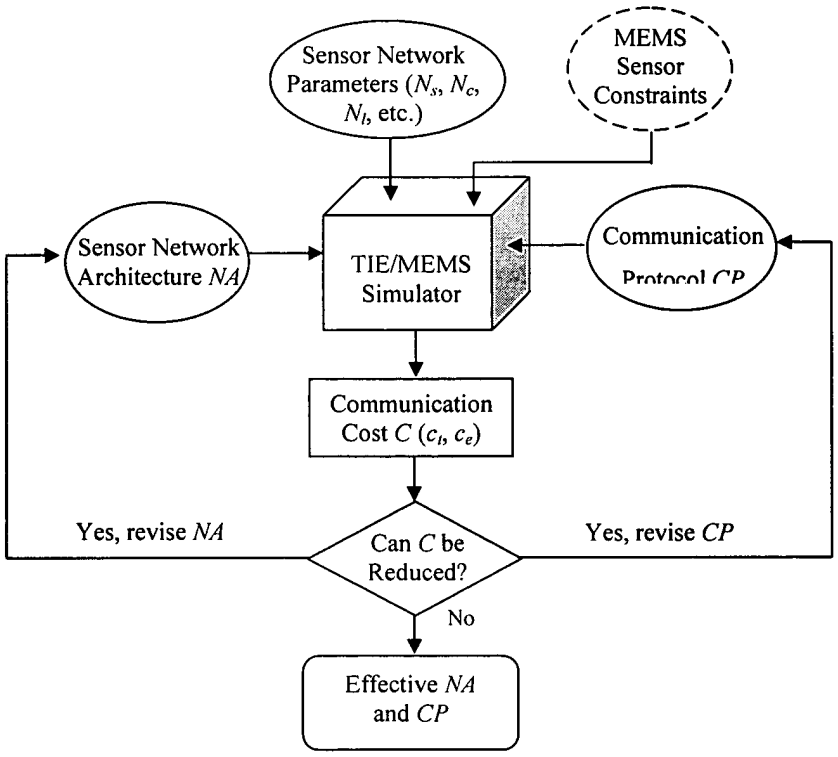


Figure 9. Structure of TIE/MEMS (dash line- under development)

166

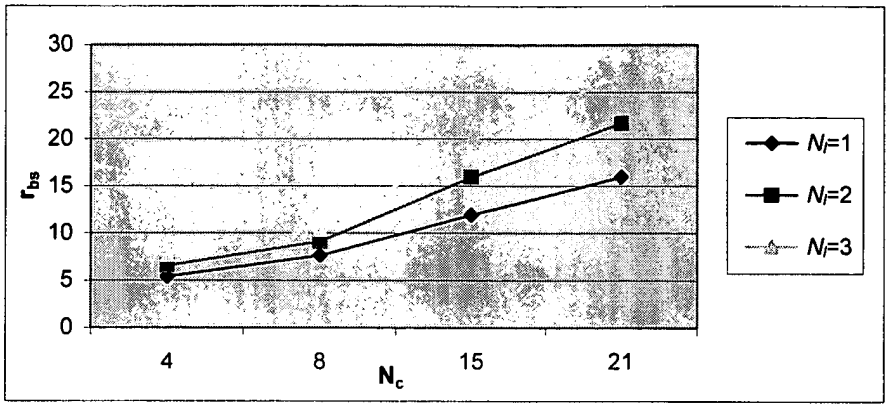


Figure 10. Ratio of Communication Time of the FTTP with Blind Base Station to the FTTP with Smart Base Station, r_{bs}, vs. No. of Sensor Nodes in the Cluster, N_c, when the No. of Failed Links, N_l, Equal to 1, 2 and 3

Fault-Tolerance in Distributed Micro Flow-Sensor Arrays and Networks

Y. Liu* and S.Y. Nof**

*School of Industrial Engineering, Purdue University, IN 47906
**Corresponding author, School of Industrial Engineering, Purdue University, IN 47906
nof@ecn.purdue.edu

Abstract: Distributed micro flow-sensor arrays and networks (DMFSA/N), built from collections of spatially scattered, cooperating intelligent micro flow-sensor nodes, can improve the accuracy and reliability of system. However, it is unrealistic to expect all the sensor nodes and communication links in the system to function properly all the time. A new fault-tolerant time-out Protocol (FTTP) with two alternatives - one with blind station, and the other with smart base station, and a fault-tolerant sensor integration algorithm (FTSIA) were developed and implemented in this research. TIE (Teamwork Integration Evaluator)/MEMS was developed to evaluate the communication protocols for the distributed micro sensor network. Experiment results show that although FTTP with smart base station requires more complex structures, compared with FTTP with blind base station, it involves less communication. Experiment results also show that FTSIA can always give reliable results even when some sensors provide faulty information, and the results from FTSIA are significantly more accurate than the statistical mean if some sensors give faulty readings.

Keywords: MEMS sensors, time-out, teamwork integration evaluator, sensor integration, communication protocol

1. Problem Background

Flow-sensors are used to measure the movement of fluid flow (liquid flow or gas flow). Conventionally, single sensor systems have been used to save space and manufacturing cost. But they are not suitable for critical applications because a single sensor is inevitably affected by noise, refresh delay and other uncertainty issues and cannot guarantee to deliver accurate information all the time. Development of MEMS (Micro Electro Mechanical Systems) has enabled production of large amounts of micro sensors at low cost. Micro flow-sensors are one of the most common MEMS devices which have been developed in recent years [1, 2, 3]. Distributed micro flow-sensor arrays and networks (DMFSA/N), built from collections of spatially scattered, cooperating intelligent micro flow-sensor nodes, can improve the accuracy, timeliness, and reliability of systems. Sensors incorporated with dedicated signal processing functions are called intelligent sensors, or smart sensors. The main roles of dedicated signal

processing are to enhance design flexibility and realize new sensing functions. Additional roles are to reduce loads on central processing units and signal transmission lines by distributing information processing to the lower layers of the system [4]. Here, the micro flow-sensor array refers to a group of the same type of micro flow-sensors which are placed close together to measure the same variable of interest. Another possibility is to combine hybrid sensors, measuring different variables, in the same array [5, 6]. Different groups of micro flow-sensor arrays can be distributed widely and selectively in the flow environment to form a micro flow-sensor network.

Due to the inherent constraints of sensors and other electronic components, and the effects of flow environment, however, it is unrealistic to expect sensor nodes and the communication links to function properly all the time. The objective of this paper is to address the design of fault-tolerant communication protocol and information integration algorithm for DMFSA/N [7]. A system is said to be fault-tolerant if its performance is not affected by faults in the system, such as sensors measure the variable of interest not accurately or timely, communication links break down, etc. A fault-tolerant time-out Protocol (FTTP) with two alternatives - one with blind stations, and the other with smart base stations, and a fault-tolerant sensor integration algorithm (FTSIA) have been proposed and implemented in this research. TIE (Teamwork Integration Evaluator)/MEMS has been developed to evaluate the communication protocols for the distributed micro sensor network.

This paper is organized as follows. In section II, the cluster architecture proposed by Liu and Nof [8], which can support fault-tolerant communication and integration for DMFSA/N, is reviewed. The new FTTP with blind base station and FTTP with smart base station are described in Section III. In section IV, TIE/MEMS is described. Two alternatives of FTTP are evaluated using TIE/MEMS. In section V, the proposed fault-tolerant sensor integration algorithm (FTSIA) is presented. Three cases of simulation experiments are conducted to test FTSIA and the results from FTSIA are compared with the statistical mean. A case study of measuring the airflow using a pressure sensor array is described in section VI to illustrate the application of FTSIA in real flow measurement. The experiment setup and results are also described. Finally, conclusions and future challenges are given in section VII.

2. A Cluster Architecture for DMFSA/N (CLA)

Since micro flow-sensor nodes are distributed spatially in DMFSA/N, a suitable architecture has to be designed to support the efficient and fault-tolerant communication and integration in the system. CLA, a cluster architecture for DMFSA/N, was proposed and developed by Liu and Nof [8]. In this structure, DMFSA/N are divided into sensor

cluster units (SCUs), each of which consists of a set of intelligent micro flow-sensor nodes and a base station. The intelligent sensor nodes within the same SCU are also called sibling nodes. An intelligent micro flow-sensor node consists of a dedicated processing unit and an associated micro flow-sensor that measures the variable of interest. A more powerful processing unit is used as the base station. The base station acts as the control node of the respective SCU, controlling the signal transmission of the sensor node, managing information rerouting in case of link failures and integrating the information from all the sensor nodes in the SCU. In order to design the system with tolerance to link failures, the sensor node sends information not only to the base station, but also to one or more other sibling nodes that can be considered as its backup nodes. Two communication protocols can be used among the sibling nodes: (1) Broadcasting Protocol (BP), in which each sensor node broadcasts its information to all its sibling nodes; (2) Evenly wide gossiping protocol (EWGP), a revised gossiping protocol (GP), in which instead of sending information to only one other sibling node by random selection according to GP [9], the backup nodes are chosen in a way that the roles of backup nodes are evenly distributed among the sibling nodes. The advantage of even distribution is to avoid a situation where some nodes are assigned too many 'duties', while others are 'starved'. These communication protocols can also be applied for the communication among base stations. The external processor is the commander of the entire network, in which information from all the SCUs is integrated for decision making as required. The clusters can be deployed along the flow path, depending on the particular flow situation and applications (figure 1). Comparison of the CLA and other architectures, including committee, hierarchical, flat tree, and multilevel binary de Bruijn architectures is described in detail in Liu and Nof [8].

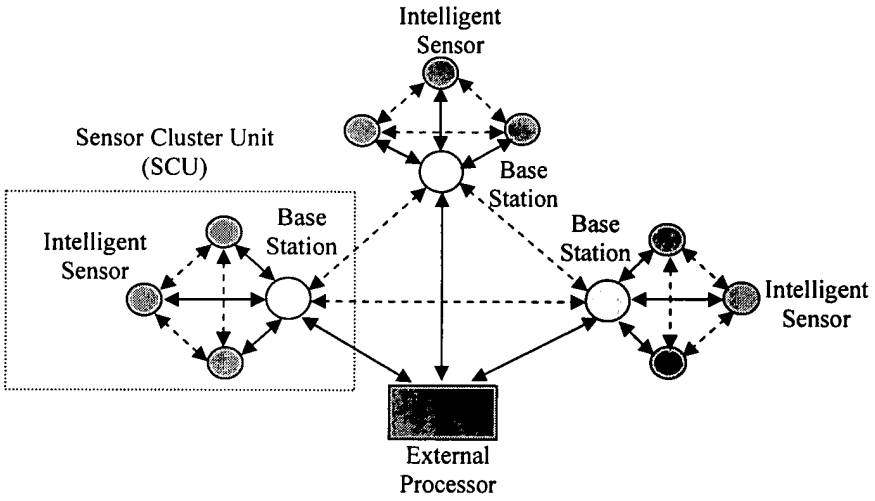


Figure 1. Proposed Cluster Architecture for DMFSA/DMFSN (dash line means the link may or may not exist depending on the communication protocol used)

III. Fault-tolerant Time-out Communication Protocol

The design of fault-tolerant time-out protocol (FTTP) is motivated by the combination of time-out and task coordination protocols [10, 11, 12, 13], and the need for fault-tolerant integration [14]. At the beginning of each data transmission cycle, the base station broadcasts the data transmission request to all the sensor nodes in the SCU. Receiving the request, each sensor node sends its data to the buffer inside the base station. Each sensor node has a corresponding buffer in the base station to store its data so that the base station can trace the source of the data. Then the data will be retrieved into the processing unit in the base station. According to FTTP logic, the base station will stop waiting for the information from a sensor node if a certain amount of time, T, has passed, where T is a pre-determined time-out period and its value which can be adaptable depends on the application. The base station will then announce the rerouting task message to its sibling nodes. Two alternatives exist for where the base station should send the rerouting task announcement: (1) with a Blind Base Station, the message is sent to all the sibling nodes; (2) with a Smart Base Station, the message is only sent to the sibling node that can finish the task earliest, based on the current status of the nodes in the system.

3.1 FTTP with Blind Base Station

In this protocol, when the link of a sensor node, say node v_k, fails, the base station will broadcast the rerouting task message (RTM) for v_k to all its sibling nodes. Receiving RTM, its backup node b_{ki} (the ith backup node of v_k) generates a bid value bv_{ki} and sends it to the base station. $b_{ki} \in BK_k$., where BK_k is the set of backup nodes of v_k. bv_{ki} can be calculated as follows:

$$bv_{ki} = (m_{ki} + 1)\gamma_{ki} \qquad (1)$$

where γ_{ki} is the transmission rate of node b_{ki} (time spent in transmitting one message) and m_{ki} is the number of tasks that have been offered to node b_{ki}. Because it is unrealistic for the intelligent sensor node to keep track of its transmission rate over time due to its computation limit, γ_{ki} is assumed to be constant for node b_{ki}. The base station then evaluates the priorities of the backup nodes based on their bid values (the lower the bid value, the higher the priority), and sends the task offer message (TOM) to the backup node with the highest priority, $b_{k(1)}$. Receiving the task offer message, $b_{k(1)}$ sends the required backup data message (BDM) to the base station. The structures of the base station and intelligent nodes, and the logic chart of the FTTP with blind base station are shown in figures 2 and 3 respectively.

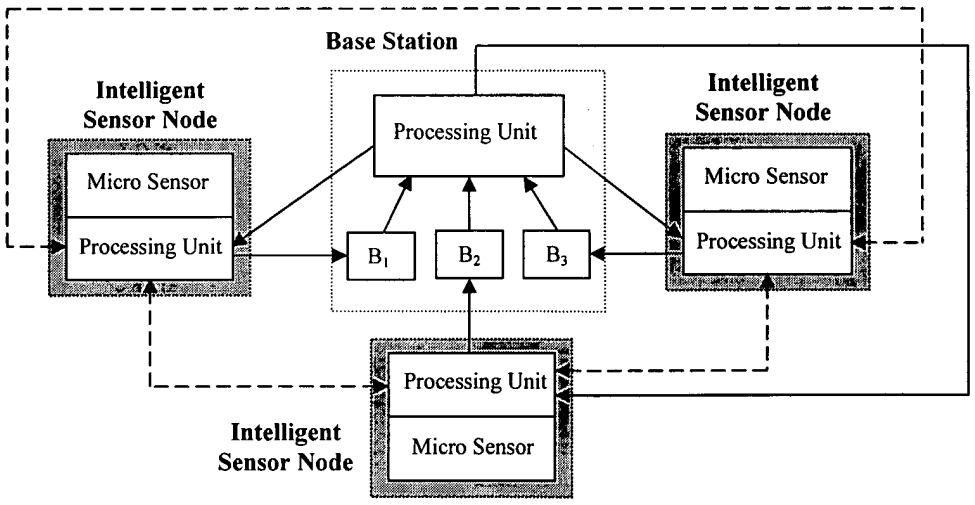


Figure 2. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Blind Base Station (Dash line means the link may or may not exist depending on the protocol used.)

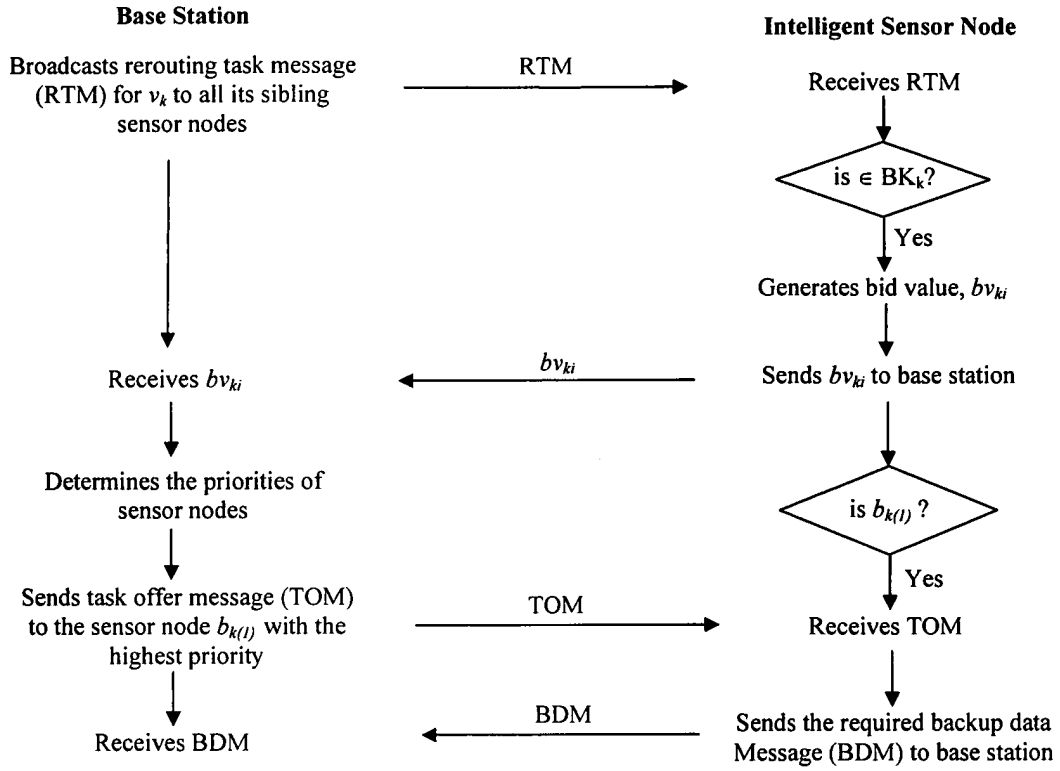


Figure 3. Logic Chart of the FTTP with Blind Base Station

3.2 FTTP with Smart Base Station

In this protocol, in order to limit the unnecessary communication and reduce the communication traffic, instead of broadcasting RTM for v_k, whose link fails, to all its sibling nodes, the base station first evaluates the priorities of the its backup nodes and then sends TOM only to the node that can finish the task the earliest. The time when the backup node b_{ki} finishes the task, t_{ki}, can be calculated as (assuming the transmission starts at time 0):

$$t_{ki} = (m_{ki} + 1)\gamma_{ki} \qquad (2)$$

where m_{ki} is the number of tasks that have been offered to the backup node b_{ki}, and γ_{ki} is the transmission rate of b_{ki} (time spent in transmitting one message). The base station then sends TOM to the node $b_{k(1)}$ whose $t_{k(1)}$ is the smallest. In order to find out t_{ki}, the base station needs to have a knowledge base which keeps track of the information for each node – its backup nodes, transmission rate, and the number of tasks that have been offered.

Figure 4 shows the structures of base station and intelligent sensor nodes for the FTTP with smart base station. The model of the knowledge base is shown in figure 5, in which n is the number of sensor nodes in the cluster and the backup nodes are hypothetical for illustration only. Figure 6 shows the logic chart of the FTTP with smart base station.

Compared with the FTTP with blind base station, the FTTP with smart base station requires less communication, and has the advantage of being able to keep track of the updated transmission rate of each sensor node, but it requires a base station with more complex structure (with an additional knowledge base). Therefore, a tradeoff has to be considered in order to determine when to apply each protocol.

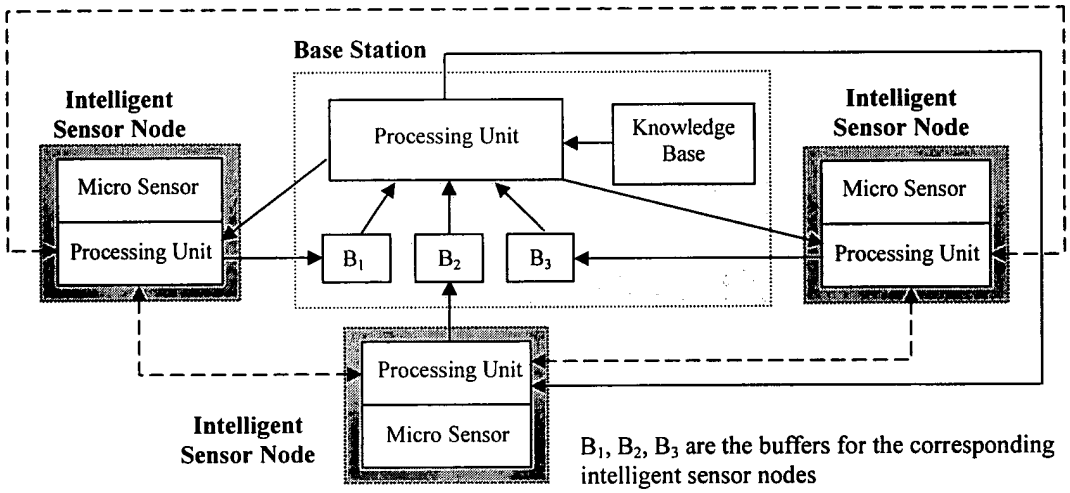


Figure 4. Structures of the Base Station and Intelligent Sensor Nodes for the FTTP with Smart Base Station (Dash line means the link may or may not exist depending on the protocol used.)

V	BK	γ	m
v_1	2, 3, 4	γ_1	m_1
v_2	1, 4, 5	γ_2	m_2
...	...	...	...
v_n	3, n-2, n-1	γ_n	m_n

Knowledge Base

Figure 5. Logic Chart of the FTTP with Smart Base Station

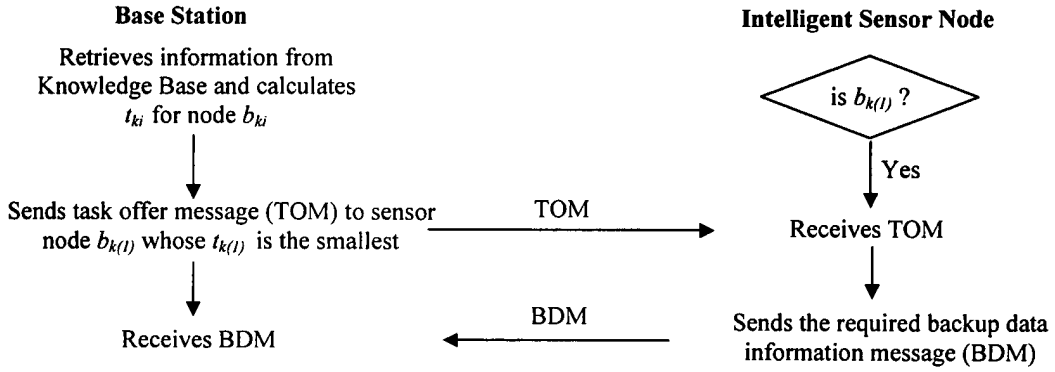


Figure 6. Logic Chart of the FTTP with Smart Base Station

4. TIE (Teamwork Integration Evaluator)/MEMS

TIE (Teamwork Integration Evaluator)/MEMS has been developed in this research as a new version of the TIE simulators developed by the PRISM group [15, 16, 17, 18] in School of Industrial Engineering at Purdue University. The objective of TIE/MEMS is to simulate and evaluate the architectures and communication protocols developed for the distributed micro sensor network.

The inputs of TIE/MEMS include micro sensor network architecture, *NA*, communication protocol, *CP*, sensor network parameters (such as the number of micro sensor nodes, N_s, the number of sensor nodes in each cluster, N_c, the number of failed links, N_l, etc.), and constraints unique to the given MEMS sensors, e.g., physical and chemical constraints causing micro-stress, proximity noise, and other disturbances to measurement and communication. The output of TIE/MEMS is the communication cost, *C*, which is represented by the communication time c_t, or/and energy consumption c_e. If *C* is too high and can be reduced, *NA* or *CP*, or both need to be revised. TIE/MEMS can be extended by adding libraries for the newly developed *NA* and *CP*. More details on TIE/MEMS can be found in Liu and Nof [18].

4.1 Simulation Experiments with FTTP

Simulation experiments were conducted to evaluate the FTTP with blind base station and FTTP with smart base station using TIE/MEMS. The experiments were implemented on the SGI Origin 2000 workstation at the Department of Computer Science in Purdue University. N_c and N_l are inputs of the experiment. Thirty experiments

were run and the output is c_t per communication round. Table 1 summarizes the experiment results, which show, as expected, that the FTTP with blind base station requires less evaluating time, c_{et}, than the FTTP with smart base station. The reason for this is that in the FTTP with blind base station, the bid values are calculated by the intelligent sensor nodes locally before they are sent to the base station, and the base station only evaluates the bid values and chooses the sensor node with the minimal bid value. Much more information transmission between the sensor nodes and the base station, however, is involved in the FTTP with blind base station. Because the information transmission time, c_{tt}, is much longer than c_{et} that requires only computation, the total communication time, c_t, which is the sum of c_{et} and c_{tt}, of the FTTP with blind base station is longer than that of the FTTP with smart base station. The difference of c_t becomes more evident when N_c and N_l increase. On the other hand, the structure of the smart base station is more complex than that of the blind base station, as shown in figure 2 and figure 4. Figure 7 shows the ratio of c_t of the FTTP with blind base station to the FTTP with smart base station, r_{bs}, vs. N_c when N_l equal to 1, 2 and 3.

Table 1. Communication Time, c_t, of the FTTP with Blind Base Station and FTTP with Smart Base Station vs. No. of Sensor Nodes in the Cluster, N_c, and No. of Failed Links, N

FTTP	N_l	N_c	c_{et}	c_{ct}	c_t
FTTP with Blind Base Station	1	4	2.17	142.77	144.93
		8	2.70	261.13	263.83
		15	2.90	477.37	480.27
		21	3.93	773.07	777.00
	2	4	3.13	346.60	349.73
		8	4.20	581.90	586.10
		15	5.87	1146.50	1152.37
		21	7.40	1706.13	1713.53
	3	4	4.63	556.80	561.43
		8	5.65	827.13	832.77
		15	7.70	1721.53	1729.23
		21	10.45	2587.52	2597.97
FTTP with Smart Base Station	1	4	4.33	22.43	26.77
		8	5.64	28.91	34.55
		15	7.16	33.06	40.23
		21	8.17	40.49	48.66
	2	4	7.90	45.33	53.23
		8	9.89	54.48	64.37
		15	13.90	58.50	72.40
		21	18.46	60.68	79.14
	3	4	10.60	62.13	72.73
		8	13.30	66.96	80.26
		15	16.83	70.69	87.51
		21	23.86	72.91	96.77

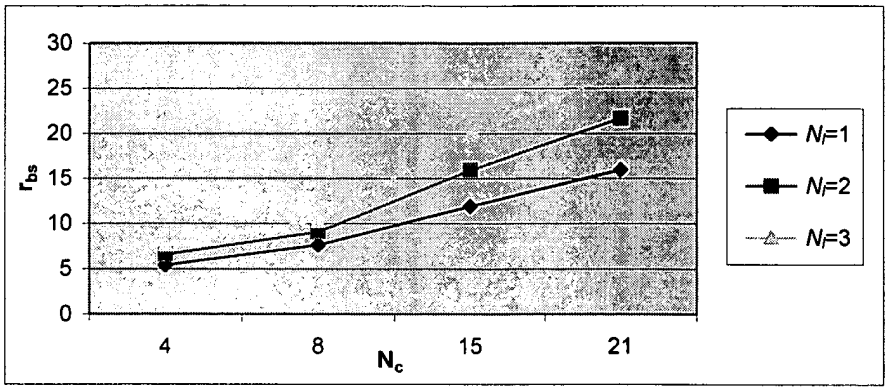


Figure 8. Ratio of Communication Time of the FTTP with Blind Base Station to the FTTP with Smart Base Station, r_{bs}, vs. No. of Sensor Nodes in the Cluster, N_c, when the No. of Failed Links, N_f, Equal to 1, 2 and 3

5. Fault-tolerant Sensor Integration Algorithm

Fault-tolerant sensor integration algorithm (FTSIA) refers to the algorithm used to combine information from different sensors in the system and produce reliable results even if some sensors yield faulty information. The algorithm developed in this research follows the ideas originated from Marzullo [19] and later improved by Jayasimha [20]. It deals with the competitive integration of sensor information, in which each sensor ideally measures identical information, but in reality, is subject to noise and other disruptions. This algorithm does not deal with the situation when sensors do not respond at all. This case is handled by the FTTP functions described earlier. First, the logic of FTSIA is described, followed by experiments demonstrating FTSIA advantages, and then explaining how FTSIA can be combined with FTTP to gain fault-tolerance advantages under time-out integration regime.

5.1 Some Definitions

A concrete sensor (CS) is a device that measures the variable of interest in the environment. An intelligent micro flow-sensor node is an example of CS. An abstract sensor (AS) is represented as a dense interval that contains the value of the physical variable read by the CS. The reason for using AS is that it is more reasonable to consider the reading from a sensor as a continuous set of values instead of a point value, because of the variation of the data due to noise, manufacturing, and other uncertainty issues. AS is derived from the point value received from CS. If a CS,

say v_i, reads a value, say τ_i, and its maximum tolerable variation can be $\pm \varepsilon_i$, then AS of v_i can be defined as the interval $[L_i, U_i] = [\tau_i - \varepsilon_i, \tau_i + \varepsilon_i]$, where L_i is the lower bound of v_i, and U_i is the higher bound of v_i.

An AS is correct if its interval contains the true value of the physical variable being measured. Otherwise, it is faulty. If AS A_1 and A_2 are correct, they must intersect and the intersection contains the true value of physical variable. An AS is said to be tamely faulty if it is not correct but its interval overlaps with that of a correct AS. A faulty AS that is not tamely faulty is said to be widely faulty.

A distinct m-interval is an interval in which no more than m intervals intersect.

5.2. The Logic of Algorithm

Input: Readings from N_s CSs, with at most f (either tamely faulty f_t, or widely faulty f_w) of them are faulty.

Output: A single output interval and a list of all the possibly tamely faulty sensors and surely faulty sensors (either tamely faulty or widely faulty).

Assumption: $f < \frac{1}{2} N_s$

Pseudo Program of FTSIA:

---

1. Import readings from N_s CSs and generate the respective ASs;

$LB =< LB_1, LB_2, ..., LB_i, ..., LB_{N_s} >$, where $LB_i = \tau_i - \varepsilon_i$;

$HB =< HB_1, HB_2, ..., HB_i, ..., HB_{N_s} >$, where $HB_i = \tau_i + \varepsilon_i$;

2. Sort the ASs by their lower bounds, and this sorted list is $L =< L_1, L_2, ..., L_{N_s} >$ and the corresponding upper bound set is $U' =< U_1', U_2', ..., U_{N_s}' >$;

3. Sort the first (N_s -f-1) intervals of U' and this sorted list is U ;

4. Initialize the variables

$I := \Phi$; /* Initialize the interval set, in which each interval has three attributes, lower bound l, higher bound u and level of interval m */

last := 0 ; /* Initialize the number of intervals in I */

5. Main part of the algorithm

for $j := f$ down to 0 do

5.1 Insert $U_{N_s - j}'$ in its appropriate place in U to keep U sorted ;

5.2 Find the intersection of ($N_s - f$) intervals

if ($L_{N_s - j} \le U_{(1)}$), then /* $U_{(1)}$ is the first element in U */

178

$N := [L_{N_s - f}, U_{(1)}]$;

5.3 Insert $N := [L_{N_s - f}, U_{(1)}]$, which is a $(N_s - f)$-interval, into I and make appropriate changes to I

5.3.1 Locate the connected interval sequence $S = [C_1, C_2, ..., C_k]$ in I in which each C_i intersects with N;

5.3.2 Condition 1: If S is empty, insert N to the tail I

if $(S = \Phi)$

$I := I \cup N$;

$last := 1$;

else

5.3.3 Condition 2: If there is only one interval C_1 is S, find the intersection of N and C_1

if $(S = [C_1])$

if $(l_{c_1} = L_{N_s - f})$

if $(U_{(1)} = u_{c_1})$ then

$m_{c_k} := m_{c_k} + 1$;

else /* if $(U_{(1)} > u_{c_1})$ */

$N_1 := [l_{c_1}, u_{c_1}]$;

$N_2 := [u_{c_1}, U_{(1)}]$;

$m_{N_1} := m_{c_1} + 1$;

$m_{N_2} := N_s - f$;

$I := (I \cup N_1 \cup N_2) - C_1$;

$last := last + 1$;

end if

else /* if $(l_{c_1} < L_{N_s - f})$ */

if $(U_{(1)} = u_{c_1})$ then

$N_1 := [l_{c_1}, L_{N_s - f}]$;

$N_2 := [L_{N_s - f}, U_{(1)}]$;

$m_{N_1} := N_s - f$;

$m_{N_2} := m_{c_1} + 1$;

$I := (I \cup N_1 \cup N_2) - C_1$;

$last := last + 1$;

else /* if $U_{(1)} > u_{c_1}$ */

$N_1 := [l_{c_1}, L_{N,-j}]$;

$N_2 := [L_{N,-j}, u_{c_1}]$;

$N_3 := [u_{c_1}, U_{(1)}]$;

$m_{N_1} := N_s - f$;

$m_{N_2} := m_{c_1} + 1$;

$m_{N_3} := N_s - f$;

$I := (I \cup N_1 \cup N_2 \cup N_3) - C_1$;

$last := last + 2$;

end if

end if

end if

else

5.3.4. Condition 3: intervals $C_2, \ldots, C_{p-1}$ completely overlaps N

for $k := 2$ to $p - 1$ do

$m_{c_k} := m_{c_k} + 1$;

end for

if($l_{c_1} = L_{N,-j}$) then

$m_{c_1} := m_{c_1} + 1$;

else /* if $l_{c_1} < L_{N,-j}$, Split $[l_{c_1}, u_{c_1}]$ into two intervals */

$N_1 := [l_{c_1}, L_{N,-j}]$;

$N_2 := [L_{N,-j}, u_{c_1}]$;

$m_{N_1} := m_{c_1}$;

$m_{N_2} := m_{c_1} + 1$;

$I := (I \cup N_1 \cup N_2) - C_1$;

$last := last + 1$;

end if

if($U_{(1)} < u_{c_p}$) then

$N_3 := [l_{c_p}, U_{(1)}]$;

$N_4 := [U_{(1)}, u_{c_p}]$;

$m_{N_3} := m_{c_p} + 1$;

$m_{N_4} := m_{c_p}$;

$I := (I \cup N_3 \cup N_4) - C_p$;

$last := last + 1$;

else /* if $U_{(1)} \geq u_{c_p}$ */

$m_{c_p} := m_{c_p} + 1$;

if ($U_{(1)} > u_{c_p}$) then

$N_5 := [u_{c_p}, U_{(1)}]$;

$m_{N_5} := N_s - f$;

$I := (I \cup N_5) - C_p$;

end if

end if

end if

5.3.5 Remove the AS $[L_{(1)}, U_{(1)}]$ from the list of ASs

$L := L - L_{(1)}; U = U - U_{(1)}$;

end for

6. Find out all the possibly tamely faulty and surely faulty sensors

for $i := 0$ up to $last - 1$ do

for $j := 0$ up to N_s do

if (($HB_j >= u_{c_i}$) or ($LB_j <= l_{c_i}$)) /* sensor j is faulty if interval i contains the true value */

$F_i = F_i \cup j$; /*insert sensor j into the set of faulty sensors of interval i; F_i is the set of faulty sensors of interval i */

end if

end for

end for

for i:=0 up to N_s

for j:=0 to last-1 do

if ($i \in F_j$) then

$n_{fi} := n_{fi} + 1$;

end if

end for

if ($n_f = 0$)

sensor i surely correct;

else

if(n_f = *last*)

sensor i is surely faulty;

else

sensor i is possibly tamely faulty;

end if

$CLB := LB - \{LB_i\}$; /*CLB is set of the lower bound of the surely correct sensors */

$CHB := HB - \{HB_i\}$; /*CHB is set of the higher bound of the surely correct sensors */

end if

end for

7. Find out the output data

sort *CLB* and the sorted list is $CL =< CL_1, CL_2, \ldots CL_m >$;

sort *CHB* and the sorted list is $CH =< CH_1, CH_2, \ldots CH_m >$;

$I_o = [CL_m, CH_1]$; /* I_o is the output interval */

5.3 Simulation Experiments and Results

Three experiment cases were simulated to test FTSIA. The mean from FTSIA, which is the average of the higher bound and lower bound of the output interval of FTSIA, were compared with the statistical mean of the same measurement values. The results from the FTSIA and statistical calculations are also affected by the distribution of sensor readings. In order to evaluate the reliability of both methods, the worst situation was simulated in all three cases as described below. All the sensor nodes generate readings above the true value of the measured physical variable, i.e., $\Delta\tau_i = \tau_i - \tau^* > 0 \ \ (i = 1,2,\ldots,N_s)$, where τ^* is the true value of the measured physical variable, τ_i is the reading received from the sensor node v_i, and $\Delta\tau_i$ is the error of the reading from v_i. The sensor readings were assumed to follow the uniform distribution. In all three cases, assuming $\tau^* = 0.5$, and the maximum tolerable variation of the sensor reading, ε, is the same for all the sensor nodes and $\varepsilon = \pm 0.005$. Shift of values from 0.5025 to 0.505 needs to be detected because the correct readings should not be greater than 0.505.

1) Case 1. In this case, all the sensors are assumed be correct, generating readings with errors within ±1% of τ^* and their ASs contain τ^*. Five experiment sets were conducted in this case: N_s is equal to 8, 15, 21, 27 and 36.

Assuming both the type I risk and type II risk are 0.025, the sample size required to detect the shift of values from 0.5025 to 0.505 is no less than seven.

2) Case 2. In this case, some sensors are assumed to be tamely faulty, i.e., they generate readings with errors beyond ±1% but within ±2% of τ^*. Since the tamely faulty ASs intersect with some correct ASs, they can form $(N_s - f)$-intervals that do not contain the true value. Because it is impossible to know in advance which interval is the correct interval, according to this algorithm, all the possibly faulty ASs are removed, and only those surely correct ASs will be kept. Following this logic, however, some correct ASs will also be considered as possibly faulty and thus removed. In order for the algorithm to be effective in this case, f_t must satisfy that $f_t < \frac{1}{2}N_s$, because there will be no correct ASs left after removing all the possibly faulty ASs when $f_t \geq \frac{1}{2}N_s$. Twelve experiment sets were conducted in this case: N_s is equal to 8 with f_t equal to 1, 2 and 3; N_s is equal to 15 with f_t equal to 1, 3, 5 and 7; and N_s is equal to 21 with f_t equal to 1, 3, 5, 7 and 10. Assuming both the type I risk and type II risk are 0.025, the sample size required to detect the shift of value from 0.5025 to 0.505 is no less than seven.

3) Case 3. In this case, some sensors are assumed to be widely faulty, i.e., they generate readings with errors beyond ±2% of τ^*. In order to control the experiments, however, their readings are assumed to be within ±10% of τ^*. Because when the readings from the widely faulty sensors are very close, it is possible that the widely faulty ASs intersect and form $(N_s - f)$-intervals that deviate the results from the correct one. In order to prevent this situation, the number of widely faulty sensors, f_w, needs to satisfy $f_w < \frac{1}{2}N_s$. Twelve experiment sets were conducted in this case: N_s is equal to 8 with f_w equal to 1, 2 and 3; N_s is equal to 15 with f_w equal to 1, 3, 5 and 7; and N_s is equal to 21 with f_w equal to 1, 3, 5, 7 and 10. Assuming both the type I risk and type II risk are 0.025, the sample size required to detect the shift of value from 0.5025 to 0.505 is no less than one.

Each experiment set was conducted twelve times. Figure 9 shows the result from FTSIA and the statistical mean vs. N_s when f=0. The results from FTSIA and statistical mean vs. f_t when N_s is equal to 8 are shown in figure 10. The results from FTSIA and statistical mean vs. f_w when N_s is equal to 8 are shown in figure 11.

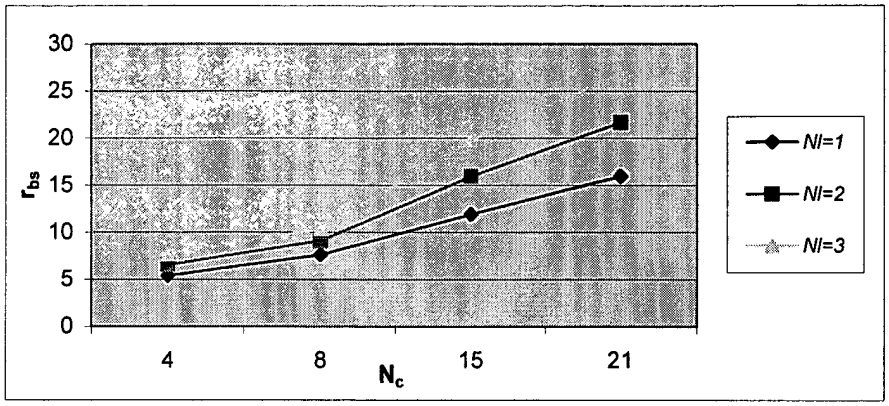


Figure 8. Ratio of Communication Time of the FTTP with Blind Base Station to the FTTP with Smart Base Station, r_{bs}, vs. No. of Sensor Nodes in the Cluster, N_c, when the No. of Failed Links, N_l, Equal to 1, 2 and 3

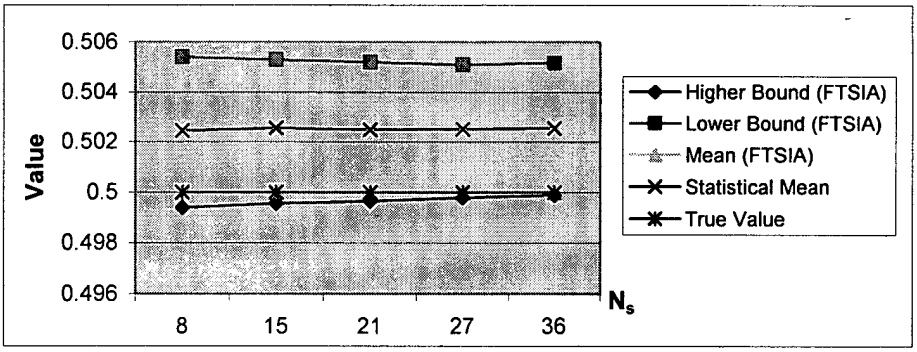


Figure 9.Result from FTSIA and Statistical Mean vs. Number of Sensor Nodes (τ^*=0.5, ε=±0.005)

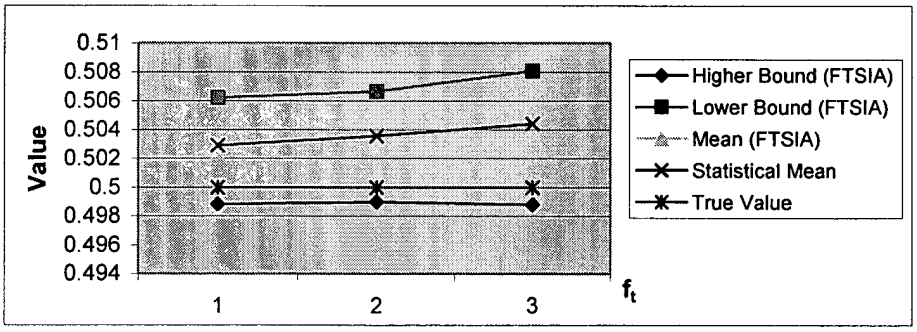


Figure 10. Result from FTSIA and Statistical Mean vs. Number of Sensor Nodes (τ^*=0.5, ε=±0.005, N_s=8)

5.4 Summary and Conclusions of Experiments

Analysis of Variance (ANOVA) was used to analyze the results from the experiments in section 5.3 (α=95%). The mean from FTSIA and statistical mean were compared. Table 2 summarizes the analysis results, which show that when f=0 or f_t is small relative to N_s, there is no significant difference between the mean from FTSIA and the statistical mean. However, when f_t increases or f_w>0, the mean from FTSIA is significantly more accurate than the statistical mean. Table 3 shows the effects of N_s and f_t on the mean from FTSIA in case 2. The reason for such effects is that tamely faulty ASs can intersect with some correct ASs and form ($N_s - f$)-intervals that do not contain the true value, and the correct ASs that do not intersect with the incorrect ($N_s - f$)-intervals will also be removed as possibly faulty ASs. The correct ASs left usually have relatively higher errors because they not only intersect with the correct ASs, but also with some tamely faulty ASs. For example, in case 2 shown in section 5.3, when N_s=15 and f_t =7, and when N_s=21 and f_t =10, only one correct ASs is left after removing all the possibly faulty ASs and this sensor has the highest sensor reading, i.e., the largest error, although still within the tolerable limits. Therefore, the performance of FTSIA degrades gracefully when f_t increases, but the output interval it generates still contains the true value. Table 3 also indicates that in order to enable FTSIA generate more accurate results, the number of correct sensors left after deletion, n_c, should be at least half of N_s, i.e., $n_c \geq \frac{1}{2} N_s$. n_c can be calculated as

$$n_c = N_s - 2 * f_t . \tag{3}$$

Therefore, when all the faulty sensors are tamely faulty in the system, in order to generate more accurate result using the FTSIA, N_s should satisfy

$$N_s \geq 4 * f_t \tag{4}$$

For example, when f_t =1 and $N_s \geq 4$, when f_t =3 and $N_s \geq 12$, and when f_t=5 and $N_s \geq 20$, the FTSIA will yield more accurate results.

The widely faulty ASs (shown in case 3), however, do not cause the same problem as tamely faulty ASs. They do not intersect with the correct ASs and thus are easier to be detected and removed, without affecting the correct ASs (see table 4). Therefore, f_w do not have significant effects on the results from FTSIA, as long as the f_w is less than half of N_s, i.e., $f_w < \frac{1}{2} N_s$. Therefore, when all the faulty sensors are widely faulty in the system, N_s should satisfy

$$N_s > 2 * f_w \tag{5}$$

Since equations (4) and (5) are linearly independent, it can be concluded that when both tamely faulty and widely faulty sensors exist in the system, in order for FTSIA to generate more accurate results, N_s should satisfy

$$N_s > 4 * f_t + 2 * f_w \qquad (6)$$

The outputs of FTSIA also indicate which sensor nodes are possibly tamely faulty, and which are surely faulty (tamely or widely). If some sensor node exists in the all the possibly faulty sensor sets, it is surely faulty. On the other hand, if some sensor node appears in some faulty sensor sets, but not all of them, it is possibly tamely faulty.

5.5 FTSIA Algorithmic Characteristics

The simulation experiment results show that the proposed FTSIA is effective because although in the simulated worst situation, when f_t is large relative to N_s, the FTSIA may degrade gracefully, it always generates the output interval that contains the true value. This algorithm has a complexity of $O(N_s \log(N_s))$ because of the sorting in step 2 and 3 shown in the logic of the algorithm (the complexity of computing statistical mean is of $O(N_s)$). This algorithm is simple and easy to be implemented.

The algorithm proposed by [19] only provides a single interval from all the sensors without fault detection. The algorithm proposed by Jayasimha [20] provides all the (N_s-i)-interval ($0 \le i \le f$), but in the flow measurement applications, single output results are preferred. The improvement of the FTSIA proposed in this thesis over the two algorithms described above is that it not only detects the possibly faulty sensors and surely faulty sensors but also generates a single output interval from the surely correct sensors after removing those faulty sensors.

Table 2. ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Faulty Sensors on the Mean from FTSIA and Statistical Mean (From three experimental cases)

f			Conclusions 1 (Mean from the FTSIA vs. Statistical Mean)	Conclusions 2 (Mean from FTSIA)
f=0			**	N_s=8 is ** from N_s=15 and ** from N_s=21**
f>0	f_i=1	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	**	
		N_s=21	**	
	f_i=3	N_s=8	*	N_s=15 is ** from N_s=21; N_s=8 is * from N_s=15 and N_s=21
		N_s=15	*	
		N_s=21	**	
	f_i=5	N_s=15	*	N_s=15 is * from N_s=21
		N_s=21	*	
	f_i=7	N_s=15	*	N_s=15 is * N_s=21
		N_s=21	*	
	f_i=10	N_s=21	*	
f_w>0	f_w=1	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	*	
		N_s=21	*	
	f_w=3	N_s=8	*	N_s=8 is ** from N_s=15 and ** from N_s=21**
		N_s=15	*	
		N_s=21	*	
	f_w=5	N_s=15	*	N_s=15 is ** from N_s=21**
		N_s=21	*	
	f_w=7	N_s=15	*	N_s=15 is ** from N_s=21**
		N_s=21	*	
	f_w=10	N_s=21	*	

* Significantly different
** Not significantly different

Table 3. ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Tamely Faulty Sensors on the Mean from FTSIA (From Case 2)

N_s	Conclusions
N_s=8	f_i=1 is ** from f_i=2; f_i=3 is * from f_i=1,2
N_s=15	f_i=1 is ** f_i=3; f_i=5 is * from f_i=1,3; f_i=7 is * from f_i=5 and from f_i=1,3;
N_s=21	f_i=1 is ** from f_i=3 and ** from f_i=5; f_i=7 is * f_i=1,3,5; f_i=10 is * from f_i=7 and * from f_i=1,3,5

* Significantly different
** Not significantly different

Table 4. ANOVA Results (α=95%) of Effects of No. of Total Sensor Nodes and No. of Widely Faulty Sensors on the Mean from FTSIA (From Case 3)

N_s	Conclusions
N_s=8	f_w=1 is ** from f_w=2 and ** from f_w=3
N_s=15	f_w=1 is ** from f_w=3 and ** from f_w=5 and ** from f_w=7
N_s=21	f_w=1 is ** from f_w=3 and ** from f_w=5 not ** from f_w=7 and ** from f_w=10

* Significantly different
** Not significantly different

6. Case Study

This section illustrates the application of the FTSIA described in section V through a case study – measuring the flow using a pressure sensor array.

6.1 Decision on the number of sensors, N_s

Before the experiment was conducted, N_s had to be decided to generate more accurate result using the FTSIA. The initial experiments showed that among the 8 available pressure sensors, f is equal to 2. Therefore, according to equation (6), all 8 available pressure sensors had to be used to form a pressure sensor array in the experiment.

6.2 Description of the experiment setup

This experiment was conducted in the PRISM lab at Purdue University. The experiment setup is shown in figure 12. It consists of a vacuum cleaner, which was used to suck air to go through the pipe connected to it, and a pressure sensor array, which had 8 pressure sensors. 8 holes were drilled on the pipe and signals were then input into the pressure sensors through the 1/8 Tygon Tubing. Because the input signals were very low (less than 0.01v), differential inputs were used to reduce the noise effect. Data Acquisition board (DAQ) was used to change the analog signals into the digital signals which were input into the computer for data analysis. In this experiment, the DAQ used was KPCI-3107, manufactured by Keithley Instruments Inc. KPCI-3107 which is compatible with LabView, etc. Finally, the data retrieved from LabView were input into the FTSIA to generate the output interval.

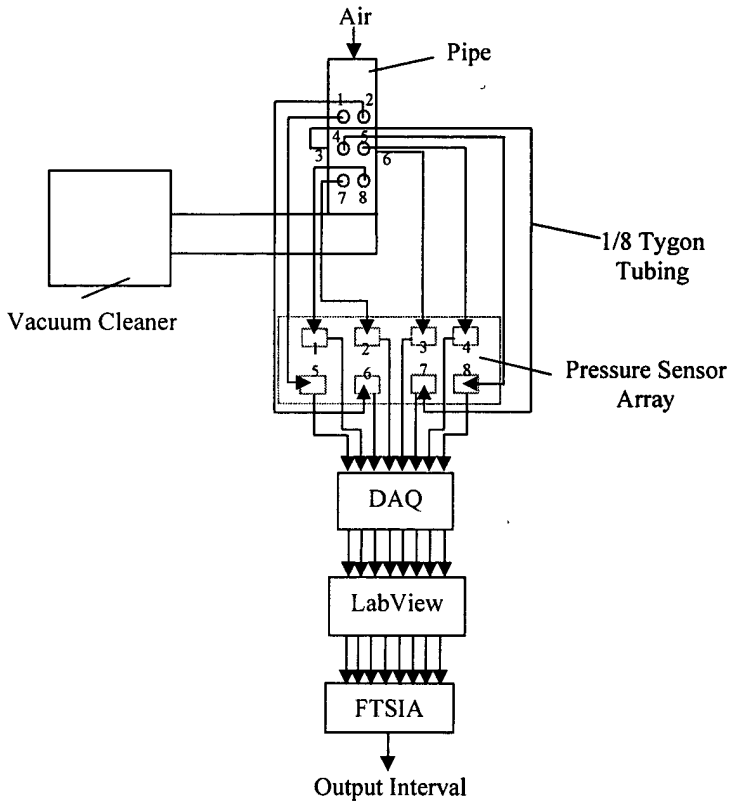


Figure 12. Experiment Setup of Flow Measurement with a Pressure Sensor Array

6.3 Experiment results

Table 5 shows an example of the data, $\tau_{i,}$, received from the pressure sensor array and the lower bounds, L_i, and higher bounds, U_i, of the data intervals, assuming ε=±0.0005. After the L_i and U_i are input into the FSTIA, the outputs generated are as shown in table 6. The output results show that there is an 8-interval, which means all ASs intersect, two 7-intervals and one 6-interval. When there are tamely faulty sensors in the system, and if the lower bounds (higher bounds) of their ASs are lower (higher) than the maximum (minimum) of the higher bounds (lower bounds) of the correct ASs, it is possible that all ASs intersect, so it cannot be concluded that there is no faulty sensor in the system. Therefore, all three possibly tamely faulty ASs, 4, 5, and 7, have to be removed, and the output interval (the intersection of all the surely correct ASs) is [0.006747, 0.007685].

Table 5. An Example of the Data Received from 8 Pressure Sensors (volt) and the Lower Bounds and Higher Bounds of the Data Interval (Assuming ε=±0.0005)

v_i	0	1	2	3	4	5	6	7
τ_i	0.007237	0.007212	0.007357	0.007158	0.007150	0.007770	0.007247	0.007132
L_i	0.006737	0.006712	0.006857	0.006658	0.00665	0.007270	0.006747	0.006632
U_i	0.007737	0.007712	0.007857	0.007658	0.00765	0.00827	0.007747	0.007632

Table 6. Summary of the Outputs from FTSIA (Data from table 5are inputs)

FTSIA	*m-Interval*					
	6-interval		7-interval		8-interval	
	Interval	Faulty Sensor(s)	Interval	Faulty Sensor(s)	Interval	Faulty Sensor(s)
	[0.007650,0.007685]	4,7	[0.006857,0.007270]	5	[0.007270,0.007632]	None
			[0.007632,0.007650]	7		
	Final Interval	[0.006857, 0.007685]		Mean	0.007271	
Statistic Calculation	Mean	0.0072829	Standard Deviation	0.0002095		

7. Conclusions and Future Tasks

This paper proposes a fault-tolerant time-out protocol (FTTP) with two alternatives and a fault-tolerant sensor integration algorithm (FTSIA) to address the fault-tolerance issues in DMFSA/N. TIE/MEMS was developed to evaluate the two alternatives of FTTP. Simulation experiments show that the FTTP with smart base station requires less communication time than FTTP with blind base station, but it demands the base station to have an additional knowledge base that keeps track of the information of the sensor nodes. Therefore, FTTP with smart base station is preferred in the on-line control applications, in which the response time is critical. Three experiment cases were simulated to test the reliability of the proposed FTSIA and the results from FTSIA were compared with the statistical mean. The experiment results show that when f_t is large relative to N_s, and when f_w >0, the FTSIA gives more accurate results than the statistical mean. The experiment results also indicate that N_s should satisfy equation (6) in order for FTSIA to give more accurate results.

The proposed FTSIA is used for integrating 1-dimensional data from the sensors that ideally should provide the same information. In the flow environment, however, the variables of interest can be a vector with more than one

dimensions. In such case, multi-dimensional fault-tolerant sensor integration algorithm needs to be developed. Moreover, it is very common that in the flow sensor network, each sensor provides partial and overlapping (complementary), or non-overlapping (cooperating) information [21] about the flow. Therefore, fault-tolerant complementary integration and cooperating integration are two other issues that will be also addressed in our future research.

Acknowledgement

This research is supported in part by Indiana 21st Century Research and Technology Fund, and by ATC Inc., a flow-sensor manufacturer. Professors John Sullivan and Steve Wereley from Purdue university and Mr. Hemi Sagi from ATC., Inc., have given useful advice.

References

1. Berberig, O., Nottmeyer, K., Mizuno, J., Kanai, Y., and Kobayashi, T., 1998, "Prandtl micro flow sensor (PMFS): A novel silicon diaphragm capacitive sensor for flow-velocity measurement", *Sensors & Actuators A-Physical.* 66(1-3) pp. 93-98.
2. Okulan, N., Henderson, H., Thurman., A., and Chong, H., 2000, "Pulsed mode micromachined flow sensor with temperature drift compensation", *IEEE Transactions on Electron Devices.* 47(2), pp. 340-347.
3. Wu, J. and Sansen, W., 2002, "Electrochemical time of flight flow sensor", *Sensors and Actuators, A: Physical.* 97-98, pp. 68-74.
4. Yamasaki, H., 1995, *Handbook of Sensors and Actuators 3, Intelligent Sensors* (Yokogawa Research Institute Corporation), Tokyo, Japan.
5. Feller, S.D., Cull, E., Kowalski, D., Farlow, K., Burchett, J., Adleman, J., Lin, C., Brady, D.J., 2002, Tracking and imaging humans on heterogeneous infrared sensor array for tactical applications, *Proceedings of SPIE - The International Society for Optical Engineering*, 4743, pp. 168-175
6. Halkyard, C R., 2003, "Sensor array design for wave decomposition in the presence of coupled motion", *Journal of Sound & Vibration.* 259(4), pp. 935-953.
7. Nof, S.Y., and Liu, Y., 2003, "Fault-tolerant Time-out Communication Protocol and Sensor Apparatus for Using Same", Patent Pending, Purdue Ref: P-02028.P1. US
8. Liu, Y., Wereley, S. T., Nof, S.Y., and Sullivan, J.P., 2001a, "Distributed micro flow-sensor array/network design and modeling". *Proceedings of PRISM Symposium* (CD-ROM), West Lafayette, IN.
9. Heinzelman, W.R., Kulik, J., and Balakrishnan, H., 1999, "Adaptive protocols for information dissemination in wireless sensor networks". *MobiCom'99, Proceedings of Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM.*, Piscataway, NJ, pp.174-185.
10. Esfarjani, k., and Nof, S.Y., 1998, "A client-server model for integration and collaboration in production testing". *International Journal of Production Research*, 36(2), pp. 3925-3321.

11. Anussornnitisam, P., Peralta, J., and Nof, S.Y., 2002, "Time-out protocol for task allocation in multi-agent systems", *Journal of Intelligent Manufacturing*, 13(6), pp. 511-522.
12. Anussornnitisam, P., and Nof, S.Y., 2003, "e-Work: the challenge for next generation ERP systems", *Production Planning & Control*, Special Issue on e-Work Model and Cases.
13. Williams, N.P., Liu, Y., and Nof, S.Y., 2002, "TestLAN approach and protocols for the integration of distributed assembly and test networks", *International Journal of Production Research*, 40(17), pp. 4505-4522.
14. Liu, Y., and Nof, S.Y, 2001b, "Distributed micro flow-sensor network design and modeling". *Proceedings of IFAC Workshop on Manufacturing, Modeling for Management and Control,* Prague, CR, pp. 161-166.
15. Khanna, N., and Nof, S.Y., 1994, "TIE, Teamwork Integration Evaluation Simulator: A preliminary User Manual for TIE 1.1". *Research Memorandum*, 94-21, School of Industrial Engineering, Purdue University, West Lafayette, IN, USA.
16. Huang, C.Y., and Nof, S.Y., 1996, "TIE: Teamwork Integration Evaluation Simulator: A manual for TIE 1.2". *Research Memorandum*, 96-2, School of Industrial Engineering, Purdue University, West Lafayette, IN, USA
17. Anussornnitisam, P., and Nof, S.Y., 2000. "a Teamwork Integration Evaluator for coordination protocols". *Proceedings of ICPR-2000* (CD-ROM), Bangkok, Thailand.
18. Liu, Y., and Nof, S. Y., 2001c, "TIE/MEMS: Modeling and Analysis of Micro Sensor Networks, User Manual". *Research Memorandum*, 01-01, School of Industrial Engineering, Purdue University, West Lafayette, IN, USA.
19. Marzullo, K., 1990, "Tolerating failures of continuous-valued sensors". *ACM Transactions on Computer Systems*, 8(4), pp.284-304.
20. Jayasimha, D.N., 1996, "Fault tolerance in multisensor networks". *IEEE Transactions on Reliability*, 45(2), pp.308-315.
21. Iyengar, S.S, Prasad, L., and Min Hla, 1995, *Advances in Distributed Sensor Technology* (Upper Saddle River, NJ, Prentice Hall PTR).

All references cited herein are incorporated in their entirety as if each were incorporated separately. This invention has been described with reference to illustrative embodiments and is not meant to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as additional embodiments of the invention, will be apparent to persons skilled in the art upon reference to this description.